University of Alberta

Dynamic Call Admission Control and Quality of Service Regulation in ATM Networks

by

Ramaswamy Srinivasan ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 1997

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-23061-9

Canada

University of Alberta

Library Release Form

Name of Author: Ramaswamy Srinivasan

Title of Thesis: Dynamic Call Admission Control and Quality of Service Regulation in ATM Networks

Degree: Doctor of Philosophy

Year this Degree Granted: 1997

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

R. Srinivasan

Ramaswamy Srinivasan
3, Matrukripa Apts,
Supermarket Complex,
Ponda, Goa,
India 403 401

Date: Oct 3, 1997

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Dynamic Call Admission Control and Quality of Service Regulation in ATM Networks** submitted by Ramaswamy Srinivasan in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dr. Paweł Gburzynski

J Harms

Dr. Janelle Harms

Wm. W. Armstrong

Dr. Bill Armstrong

Ron Unrau

Dr. Ron Unrau

Dr. Horacio Marquez

Dr. Jim Griffioen

Date: Sept. 30, 1997

To

My dear Mother and Father

for their support, encouragement and affection

Contents

1 Introduction **1**
1.1 Background 1
1.2 Scope 3
1.3 Thesis Contribution 3
1.4 Outline 5
1.5 List of publications 6

2 Simulation-based effective bandwidth characterization **8**
2.1 Introduction 8
2.2 Notation 12
2.3 Experimental setup 13
2.3.1 Traffic model 13
2.3.2 Factors affecting cell delay and loss rate 13
2.3.3 Training set data 15
2.3.4 Test set data 17
2.4 Adaptive logic networks 18
2.4.1 Choosing the epsilon values 20
2.4.2 ALN model for cell delay 21
2.4.3 ALN model for cell loss rate 23
2.4.4 Residual analysis for ALN models 25
2.5 Regression analysis 27
2.5.1 Regression model for delay 28
2.5.2 Regression model for cell loss rate 32
2.5.3 Residual analysis for regression models 37
2.6 The Anick-Mitra-Sondhi model 38
2.6.1 AMS cell loss rate model 38
2.6.2 Extended AMS cell delay model 40
2.7 Comparison of cell delay and loss rate predictions 41
2.7.1 Cell delay based comparisons 41
2.7.2 Cell loss rate based comparisons 43
2.8 Application to effective bandwidth prediction 43
2.8.1 Effective bandwidth vs. number of calls 44
2.8.2 Effective bandwidth vs. buffer size 45
2.9 Conclusions 46

3 Cell scheduling using adaptive feedback control **49**
3.1 Introduction . . . 49
3.2 Motivation . . . 49
3.3 Background . . . 51
3.3.1 Feedback control . . . 52
3.3.2 System identification . . . 53
3.3.2.1 Modeling the input-output relationship . . . 55
3.3.2.2 Batch model identification . . . 56
3.3.2.3 Online model identification . . . 59
3.3.2.3.1 Initializing the recursive estimator . . . 60
3.3.2.3.2 Adaptive RLS . . . 61
3.3.2.3.3 Rigorous recursive least squares (RRLS) estimation . . . 62
3.3.3 Controller design . . . 63
3.3.4 Adaptive control . . . 65
3.4 The queuing system . . . 67
3.4.1 Source traffic descriptor . . . 67
3.4.2 Quality of service . . . 68
3.5 System model . . . 70
3.5.1 Choosing the plant output signal . . . 70
3.5.1.1 Measuring the buffer quantile . . . 71
3.5.1.2 Calculating the buffer quantile . . . 72
3.5.2 Calculation of the reference signal . . . 76
3.5.3 Choosing the plant input signal . . . 76
3.6 Traffic models . . . 77
3.6.1 Poisson traffic . . . 77
3.6.2 Batch-Poisson traffic . . . 78
3.6.3 Self-similar traffic . . . 78
3.6.3.1 Self-similar traffic generator . . . 81
3.7 Open-loop identification . . . 82
3.7.1 Choosing the plant sampling time . . . 86
3.8 Closed-loop control . . . 86
3.8.1 Choosing the controller sampling time . . . 90
3.8.2 Service rate constraints . . . 91
3.9 Results . . . 97
3.9.1 Self-tuning controller . . . 98
3.9.1.1 Poisson traffic . . . 100
3.9.1.2 Batch-Poisson traffic . . . 110
3.9.1.3 Self-similar traffic . . . 120
3.9.2 Adaptive controller . . . 130
3.9.3 Effect of forgetting factor . . . 131
3.9.4 Effect of mean service rate . . . 135
3.10 Conclusions . . . 137

4 Dynamic call admission control 140
4.1 Introduction 140
4.2 System model 142
4.3 Bandwidth-buffer tradeoff 142
4.4 Dynamic CAC 149
4.4.1 The S-CAC algorithm 150
4.4.2 The D-CAC algorithm 151
4.4.2.1 D-CAC on call departure 153
4.5 Results 153
4.5.1 Experiment setup 153
4.5.2 S-CAC : effect of factors on VBR call blocking 155
4.5.3 D-CAC : effect of factors on VBR call blocking 157
4.5.4 The modified D-CAC algorithm 160
4.6 Conclusions 165

5 Conclusions 166

6 Future Work 168

Bibliography 170

List of Figures

2.1 System model 9
2.2 Source model 14
2.3 Delay vs. N (left) Delay vs. S (right) 16
2.4 Delay vs. Eb (left) Loss rate vs. Eb (right) 17
2.5 D-ALN model predictions: training set (left) test set (right) 23
2.6 C-ALN model predictions: training set (left) test set (right) 24
2.7 Extrapolation for N=30: D-ALN model (left) C-ALN model (right) . . . 25
2.8 Histogram of residuals: D-ALN model (left) C-ALN model (right) . . . 25
2.9 Plot of residuals: D-ALN (left) C-ALN (right) 26
2.10 D-REG model predictions: training set (left) test set (right) 33
2.11 C-REG model predictions: training set (left) test set (right) 36
2.12 Extrapolation for N=30: D-REG model (left) C-REG model (right) . . 36
2.13 Plot of residuals: D-REG model (left) C-REG model (right) 37
2.14 Histogram of residuals: D-REG model (left) C-REG model (right) . . . 38
2.15 C-AMS predictions: training set (left) test set (right) 39
2.16 Infinite buffer delay and conditional delay 41
2.17 Infinite buffer jitter and conditional jitter 41
2.18 D-AMS predictions: training set (left) test set (right) 42
2.19 Effective bandwidth vs. number of calls 45
2.20 Effective bandwidth vs. buffer size 46

3.1 Feedback control system 52
3.2 Open-loop system identification 53
3.3 Closed-loop system identification 54
3.4 System representation : Filter (left) Transfer function (right) 55
3.5 Adaptive feedback controller 66
3.6 ATM Forum CTD probability density model 68
3.7 Queuing system 70
3.8 Buffer quantile histogram 71
3.9 Feedback control of calculated quantile 74
3.10 Batch Poisson traffic model 78
3.11 Open loop steady-state response 83
3.12 Open loop step response 84
3.13 Open loop identification 85
3.14 SIMULINK feedback control system (modified from [19]) 87
3.15 Feedback control - static constraints 92

3.16 Self-tuning control (Poisson traffic, $\rho = 0.8$, seed = 1) 101
3.17 Self-tuning control (Poisson traffic, $\rho = 0.8$, seed = 2) 102
3.18 Self-tuning control (Poisson traffic, $\rho = 0.8$, seed = 3) 103
3.19 Self-tuning control (Poisson traffic, $\rho = 0.75$, seed = 1) 104
3.20 Self-tuning control (Poisson traffic, $\rho = 0.75$, seed = 2) 105
3.21 Self-tuning control (Poisson traffic, $\rho = 0.75$, seed = 3) 106
3.22 Self-tuning control (Poisson traffic, $\rho = 0.6$, seed = 1) 107
3.23 Self-tuning control (Poisson traffic, $\rho = 0.6$, seed = 2) 108
3.24 Self-tuning control (Poisson traffic, $\rho = 0.6$, seed = 3) 109
3.25 Self-tuning control (Batch Poisson traffic, $\rho = 0.8$, seed = 1) 111
3.26 Self-tuning control (Batch Poisson traffic, $\rho = 0.8$, seed = 2) 112
3.27 Self-tuning control (Batch Poisson traffic, $\rho = 0.8$, seed = 3) 113
3.28 Self-tuning control (Batch Poisson traffic, $\rho = 0.75$, seed = 1) 114
3.29 Self-tuning control (Batch Poisson traffic, $\rho = 0.75$, seed = 2) 115
3.30 Self-tuning control (Batch Poisson traffic, $\rho = 0.75$, seed = 3) 116
3.31 Self-tuning control (Batch Poisson traffic, $\rho = 0.6$, seed = 1) 117
3.32 Self-tuning control (Batch Poisson traffic, $\rho = 0.6$, seed = 2) 118
3.33 Self-tuning control (Batch Poisson traffic, $\rho = 0.6$, seed = 3) 119
3.34 Self-tuning control (self-similar traffic, $\rho = 0.8$, seed = 1) 121
3.35 Self-tuning control (self-similar traffic, $\rho = 0.8$, seed = 2) 122
3.36 Self-tuning control (self-similar traffic, $\rho = 0.8$, seed = 3) 123
3.37 Self-tuning control (self-similar traffic, $\rho = 0.75$, seed = 1) 124
3.38 Self-tuning control (self-similar traffic, $\rho = 0.75$, seed = 2) 125
3.39 Self-tuning control (self-similar traffic, $\rho = 0.75$, seed = 3) 126
3.40 Self-tuning control (self-similar traffic, $\rho = 0.6$, seed = 1) 127
3.41 Self-tuning control (self-similar traffic, $\rho = 0.6$, seed = 2) 128
3.42 Self-tuning control (self-similar traffic, $\rho = 0.6$, seed = 3) 129
3.43 Open loop test (varying parameters) 130
3.44 Adaptive control (effect of varying parameters) 131
3.45 Open loop test (varying parameters) 132
3.46 Adaptive control (effect of varying parameters) 132
3.47 Self-tuning control (effect of forgetting factor, $\rho = 0.8$, FF = 1) . . . 133
3.48 Self-tuning control (effect of forgetting factor, $\rho = 0.8$, FF = 0.95) . . 133
3.49 Self-tuning control (effect of forgetting factor, $\rho = 0.8$, FF = 0.9) . . 133
3.50 Self-tuning control (effect of forgetting factor, $\rho = 0.75$, FF = 1) . . . 134
3.51 Self-tuning control (effect of forgetting factor, $\rho = 0.75$, FF = 0.95) . 134
3.52 Self-tuning control (effect of forgetting factor, $\rho = 0.75$, FF = 0.9) . . 135
3.53 Self-tuning control (effect of mean service rate, $\rho = 0.6$) 136
3.54 Self-tuning control (effect of mean service rate, $\rho = 0.8$) 136
3.55 Effect of mean service rate on buffer occupancy CDF 137

4.1 System model 142
4.2 Hypothetical resource tradeoff plot for a VBR source 143
4.3 Tradeoff plot : regression (left) ALN (right) 145
4.4 Tradeoff plot : regression (left) ALN (right) 146

4.5 Tradeoff plot : regression (left) ALN (right) 147
4.6 Tradeoff plot : regression (left) ALN (right) 147
4.7 Tradeoff plot for a VBR call (Fixed buffer size) 150
4.8 D-CAC - incoming VBR call 152
4.9 D-CAC - departing VBR/ABR call 153
4.10 S-CAC : effect of Eb/L 155
4.11 S-CAC : effect of Eb/Pcr 156
4.12 S-CAC : effect of ρ_{abr}/ρ_{vbr} 157
4.13 S-CAC : effect of ρ_{abr}/ρ_{vbr} 157
4.14 D-CAC : effect of Eb/L 158
4.15 D-CAC : effect of Eb/Pcr 159
4.16 D-CAC : effect of ρ_{abr}/ρ_{vbr} 159
4.17 D-CAC : effect of ρ_{abr}/ρ_{vbr} 159
4.18 MD-CAC : effect of α^{max}_{ABR} 161
4.19 MD-CAC : effect of α^{max}_{ABR} 162
4.20 MD-CAC : achieved ABR blocking rate 162
4.21 MD-CAC : achieved ABR blocking rate 163
4.22 MD-CAC : dynamic gain 163
4.23 MD-CAC : dynamic gain 164

List of Tables

2.1 Levels of factors for training set 16
2.2 Levels of factors for test set 17
2.3 Epsilons for D-ALN and C-ALN 20
2.4 Choosing the output epsilon for D-ALN 22
2.5 Choosing rms_{min} for D-ALN 23
2.6 CP criterion for D-REG sub-model (stage 1) 30
2.7 *CP* criterion for D-REG sub-model (stage 2) 31
2.8 *CP* criterion for D-REG sub-model (stage 3) 32
2.9 *CP* criterion for C-REG sub-model (stage 1) 34
2.10 *CP* criterion for C-REG sub-model (stage 2) 35
2.11 *CP* criterion for C-REG sub-model (stage 3) 35
2.12 Numerical comparison of delay predictions 42
2.13 Numerical comparison of cell loss predictions 43

4.1 QoS levels for tradeoff plots 145
4.2 Bellcore VBR test source parameters 154
4.3 Levels of factors used in experiments 155
4.4 D-CAC v/s S-CAC 160

Abstract

Call admission control (CAC) is an important mechanism to prevent congestion in ATM networks and to ensure that admitted connections receive the quality of service promised at call setup time. In order to ensure high network utilization, CAC must be able to quickly and acurately predict the switch resources required by a new call.

In this thesis, we develop simulation-based models—using neural network modeling and regression analysis techniques—that predict the bandwidth requirement of a group of statistically multiplexed bursty calls with real-time quality of service requirements. We demonstrate that our models are more accurate than a well-known analytical model and yet require less computation. We argue that the increasing complexity of ATM traffic makes it difficult or even impossible to obtain reasonably accurate analytical models. Therefore, the relative ease with which the neural network model can be developed makes it a viable alternative for bandwidth prediction for use in real-time CAC.

The quality of service actually provided to the aggregate traffic may vary about the desired level—even when the bandwidth allocated by CAC is highly accurate—as a result of bit-rate fluctuations and because individual sources may not utilize the entire bandwidth requested by them. We demonstrate through the use of an adaptive feedback control scheduler that we are able to maintain the quality of service at a constant level in spite of traffic fluctuations and inaccurate service rate allocation by CAC. This results in better network utilization.

Since real-time traffic has stringent quality of service constraints, we argue that real-time service would be more expensive as compared to best-effort service. In such a situation, the network provider can increase revenue by giving priority to real-time connections at call admission time. We propose a dynamic CAC that lowers the call blocking rate for real-time traffic at the expense of a higher blocking rate for best-effort traffic. The impact of the scheme on best-effort call blocking rate can be controlled by the network provider, allowing an increase in revenue to be weighed against an increase in the number of dissatisfied best-effort service customers. We show that the dynamic scheme performs very well for bursty calls whose bandwidth requirement is a small fraction of the link rate.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Pawel Gburzynski, for accepting me as his graduate student, and for the support and encouragement I have received these past years. I appreciate the freedom that he gave me, as well as the help that he unhesitatingly offered throughout my graduate period.

I would like to thank Dr. Janelle Harms for the great deal of help that I have received, particularly for readily giving up her valuable time for discussion of my ideas. The papers and books that she made available to me were very useful in my research. Sincere thanks are due to Dr. Bill Armstrong for providing me with the ALN software, and for his patience in dealing with questions arising from my use of it. I would like to thank Neal Sanche for making the Unix version of the software, developed by him for Dendronic Decisions Ltd., available to me.

I am indebted to Dr. S. Lakshminarayan for his advice and help in implementing the feedback controller. Dr. Shah, of the Department of Chemical Engineering, was very encouraging of my efforts to utilize feedback control theory, for which I thank him deeply. I would like to thank Dr. Danyang Liu, also of the Department of Chemical Engineering, for providing me with the source code of his feedback controller, and for his ready help when I ran into difficulty.

I would like to express my gratitude to Mrs. Lakshmi Ramanan, for her efforts to make me feel at home when I first came here five years ago, and for her selfless help over this period. The companionship of my good friends Dr. Lakshminarayan, Dr. Ravindra Gudi, Dr. Ashish Paradkar and Kaladhar Voruganti made my stay in Edmonton very pleasant.

Thanks are due to my colleagues, Theodore Ono-Tesfaye and Dr. Michael Boshra, for their help, as well as for the many interesting technical (and otherwise) discussions. A big thank-you to Anne Nield for all her help.

I would like to thank my dear parents for their support and encouragement; without their sacrifices, I would not be here today. I would also like to thank my in-laws, sister, brother-in-law and sisters-in-law for their love and support over the years.

Last, but not least, I would like to thank my dear wife, Manisha; her support and companionship made the process of obtaining a Ph.D more enjoyable than it ought to be.

Chapter 1

Introduction

1.1 Background

Broadband integrated service digital networks (B-ISDN) are designed to carry traffic of several different classes, ranging from voice and interactive data to still-picture and full-motion video. Asynchronous Transfer Mode (ATM) is the proposed transport[1] technology for deployment in B-ISDNs. It provides a unified interface for use by services with drastically different quality of service (QoS) requirements.

Traffic in an ATM network is carried in small, fixed size packets known as *cells*. A connection must be set up between a source and destination and resources reserved for the call along the route before the start of data transfer—this notion of connection-oriented service is borrowed from the realm of circuit-switched telephone networks. Thus, we use the term "call" to refer to a connection.

A call requesting admission to the network provides information called a traffic descriptor that specifies the expected behavior of the traffic that will flow over the connection. It also indicates the QoS required for the duration of the connection. Based on this information, the call admission control (CAC) at each switch on the route between the connection endpoints reserves bandwidth and cell buffers for the connection. If the reservation fails at any switch on the route due to insufficient resources, alternative routes not passing through that switch may be tried. If no end-to-end path is available with adequate resources, the call is refused admission to the network. Once a connection is admitted, it (ideally) cannot be torn down by the network, or have its QoS reduced[2].

---

[1] "Transport" refers to the use of ATM switching and multiplexing techniques at the data-link layer of the ISO OSI model

CAC deals with the question of whether or not a switch can accept a new connection. A new connection can be accepted if:

- The new connection does not affect the QoS of existing connections
- The switch has adequate resources (bandwidth, buffers) to provide the QoS required by the new call.

If the switch admits connections indiscriminately, the network may become congested and unable to meet the performance objectives of existing connections. By admitting only those calls that do not adversely affect calls already admitted, CAC acts as a preventive congestion control.

CAC must also be efficient. If a call is turned away because the resources required were estimated inaccurately by CAC, revenue will be lost and customers may take their business elsewhere. For this reason, CAC must be accurately able to determine the bandwidth and buffer requirement of an incoming call.

Allocating peak bandwidth (non-statistical bandwidth allocation) is inefficient when the traffic generated by a call is bursty. In statistical allocation, the bandwidth allocated to a group of connections can be less than the sum of their peak rates and yet be sufficient to satisfy the QoS requirement. This bandwidth is a function of several parameters: the number of available cell buffers, the traffic descriptor of each call and its QoS requirement. It may also be a function of the number of calls in the group, assuming that traffic from the calls is statistically multiplexed on the output link and shares link bandwidth.

Statistical allocation makes economic sense when allocating resources to a number of bursty calls. However, due to difficulty in characterizing the aggregate traffic of multiplexed bursty calls, and because of the complicated relationship between the required bandwidth and the call parameters, it is difficult to determine the bandwidth accurately. CAC decisions must be made quickly. Therefore, the decision algorithm cannot be too complex.

---

[2] While this may not always be true, we assume this is the case in this thesis.

1.2 Scope

We study the problem of characterizing the bandwidth requirement of a group of variable bit rate (VBR) calls multiplexed together over an output link of an ATM switch. Available bit rate (ABR) traffic sharing the output link with the VBR traffic is assumed to be multiplexed in a separate queue.

A weighted fair queuing (WFQ)[10] or similar scheduler is assumed to be employed at the output port. This scheduler guarantees a minimum service rate to each of the two queues, the minimum rates being determined by CAC. Unused VBR bandwidth is assumed to be absorbed by the ABR traffic, since ABR sources are assumed to be "greedy".

VBR calls are assumed to be homogeneous (that is, they have identical traffic descriptors) and have the same QoS requirement. We consider only real-time VBR calls (ATM Forum [8] **rt-vbr** service class). The per-node cell delay and delay jitter bounds are assumed to be derivable from the corresponding end-to-end specifications. All ABR calls are assumed to have the same minimum bandwidth requirement.

Because VBR calls have a more stringent QoS requirement as compared to ABR calls, we assume that the tariff for a VBR call would be higher than for a ABR call with identical source behavior.

1.3 Thesis Contribution

- The call admission control schemes described in the literature (chapter 2) are based on analytical solution of a queuing system, the input to which is generated by a number of sources modeled individually (each source could be assumed to be an Interrupted Poisson Process, for example), or by an aggregate source (frequently modeled by a Markov-Modulated Poisson Process). Simplifications are necessary to derive a solution; a common assumption, for example, is that the cell loss rate can be approximated by the buffer overflow probability. The solution, which is time-consuming to develop, can be too computationally-intensive for use in real-time CAC, for a desired level of accuracy.

We therefore investigate the possibility of using data-mining techniques such as regression analysis and neural networks for bandwidth prediction. While the scope of applicability of these models is not as wide as that of analytical solutions, particularly at very low cell loss rates for which the data may be difficult to come by; we show that they are capable of being far more accurate while being simple to compute. This can improve CAC efficiency and lead to lower call blocking rates.

- Even if the actual traffic generated by a source conforms to the traffic specification given at call admission time, traffic fluctuations may cause QoS variations and the observed QoS may occasionally violate the requirement. Low frequency variations in the parameters of a conforming source over time may also cause QoS fluctuation. When the source generates less traffic than specified, the higher than necessary service rate will result in a higher QoS than required. The extra bandwidth could be better utilized by ABR traffic.

  For this reason, we modify the WFQ scheduler so that the bandwidth allocated to the VBR queue is dynamically varied in such a way that the observed VBR QoS always remains at the desired level, in spite of VBR traffic fluctuations and individual source parameter variations. We demonstrate the effectiveness of the scheme in controlling the QoS for three very different aggregate traffics.

- As a consequence of the assumption (section 1.2) that VBR calls will be charged a higher price by the network provider, it follows that for the same volume of ABR and VBR traffic, the latter will generate a greater revenue. On the other hand, if both types of traffic compete at the same priority at call admission time, as is the case in current CAC schemes, it could result in rejection of VBR calls as a result of admitting ABR calls.

We therefore propose a call admission scheme which lowers the call blocking rate for VBR to the maximum extent possible. The maximum acceptable blocking rate for ABR traffic can be specified. Our scheme tries to favor VBR as long as this upper bound is not reached. We show that the scheme is easy to implement and can be of considerable benefit to VBR calls.

We use the term *dynamic* CAC to refer to this scheme since the bandwidth allocated to the VBR queue is a function of the number of ABR and VBR calls currently admitted at the switch. Hence, the bandwidth allocation is non-deterministic.

In the literature however, CAC schemes are described as *dynamic* if the traffic descriptors used in decision making are derived from measurements as opposed to being statically declared by the source. This improves the efficiency of CAC[3].

1.4 Outline

Chapter 2 describes the regression and neural network models and compares their performance with the analytical Anick-Mitra-Sondhi [4] model. Chapter 3 describes the modified WFQ scheduler implemented using adaptive feedback control. In chapter 4, we evaluate the performance of our dynamic CAC scheme and compare it with a static control. Chapter 5 sums up the results and chapter 6 describes extensions to this work that may be considered.

---

[3] However, the QoS is still subject to variation due to traffic fluctuation and source parameter changes, as in static CAC

1.5 List of publications

1. P. Gburzynski, T. Ono-Tesfaye, and S. Ramaswamy. Modelling ATM networks in a parallel simulation environment: a case study. *Summer Computer Simulation Conference (SCSC) '95*, Ottawa, Canada, July 24–26, 1995.

2. S. Ramaswamy, T. Ono-Tesfaye, W. W. Armstrong and P. Gburzynski. A simulation approach to equivalent bandwidth characterization in ATM networks. *International Conference on Modelling, Simulation and Optimization (IASTED) '96*, Gold Coast, Australia, May 1996.

3. T. Ono-Tesfaye, S. Ramaswamy and P. Gburzynski. Effective bandwidth for bounded cell loss, delay and jitter. *IEEE ATM '96 Workshop*, San Francisco, August 1996.

4. S. Ramaswamy and P. Gburzynski. An adaptive feedback control based cell scheduler for ATM networks. *IEEE Workshop on Resource Allocation Problems in Multimedia Systems*, Washington, D.C., December 1996.

5. S. Ramaswamy, T. Ono-Tesfaye and P. Gburzynski. A regression approach to equivalent bandwidth characterization in ATM networks. *Proceedings of the Fifth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) '97* , Haifa, Israel, January 12–15, 1997.

6. S. Ramaswamy and P. Gburzynski. A neural network approach to equivalent bandwidth characterization in ATM networks. *Fifth IFIP Workshop on Peformance, Modelling, and Evaluation of ATM Networks*, Ilkley, West Yorkshire, U.K., July 21–23, 1997.

7. S. Ramaswamy, T. Ono-Tesfaye, W. W. Armstrong and P. Gburzynski. Equivalent bandwidth characterization for real-time CAC in ATM Networks. *Sub-*

mitted to Journal of High Speed Networks, 1997.

8. S. Ramaswamy, P. Gburzynski and D. Liu. ATM cell scheduling using adaptive feedback control. *Submitted to Journal of High Speed Networks*, 1997.

9. S. Ramaswamy and P. Gburzynski. A dynamic call admission control scheme for VBR traffic in ATM networks. *Submitted to INFOCOM '98.*

Chapter 2

Simulation-based effective bandwidth characterization

2.1 Introduction

High-speed ATM-based integrated networks are expected to carry connections with a wide range of characteristics. The call admission control mechanism in such networks tries to avoid congestion by limiting the number of connections that are admitted into the network. Many call admission schemes are based on the concept of *effective bandwidth* (EB) or equivalent capacity: the effective bandwidth of a traffic source is defined as the bandwidth required to guarantee some specific quality of service (QoS). In such a scheme, a call is admitted by a switch if the effective bandwidth of the call is less than or equal to the bandwidth still available on the outgoing link. The effective bandwidth of a source lies between its average and peak cell rates. The effective bandwidth of N aggregated sources is generally less than N times the effective bandwidth of a single source (this phenomenon is known as *statistical multiplexing gain*).

Most of the effective bandwidth research has focussed on the cell loss rate (CLR) as the exclusive QoS metric. The queuing system shown in figure 2.1 is assumed to be one of several queues at an output port of an ATM switch. Link bandwidth is shared among the queues[1]. Therefore, the service capacity of the queue is a fraction

---

[1] One of the ways in which the traffic destined for the outgoing link may be partitioned among

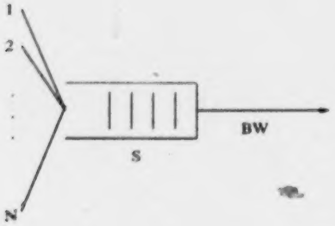


Figure 2.1: System model

of the total link capacity.

If the CLR for N sources multiplexed at the queue can be determined as a function of the buffer size S, the service rate (or bandwidth) Bw and other system parameters,

$$CLR = f(S, Bw, \ldots) \tag{2.1}$$

the effective bandwidth for N sources can be determined by either explicitly inverting the above function or by iteration. In most cases, unfortunately, a function like (2.1) is not explicitly available, and approximations must be used. Anick, Mitra and Sondhi [4] present a well-known ***fluid-flow*** model (hereafter referred to as the AMS model) for the case of N identical On/Off sources feeding an infinite buffer. Because of the infinite buffer assumption, the cell loss rate for a finite buffer of size S is approximated by the probability $P(x > S)$ that the occupancy of the infinite buffer exceeds S. Guerin et al. [15] propose a simplified, computationally tractable version of the AMS model. A simple ***binomial*** approach is described in [21]: the authors' proposal is to approximate the cell loss rate by the probability that the combined cell rate of all N sources exceeds the link capacity (in effect, this assumes zero cell buffers). A survey of call admission control schemes proposed in the literature is presented in [27]. Various effective bandwidth schemes are compared in [31, 34].

The limitations of these analytical models are:

1. They require the QoS requirement to be expressed in terms of the buffer overflow probability, rather than the cell loss probability. For low to moderate buffer

---

the various queues is according to traffic class.

sizes, the difference in estimated bandwidth requirement can be significant.

2. They consider only the cell loss rate as a performance metric. In order to limit average cell delay as well as loss rate, however, it is necessary to base EB computations on estimates of the cell delay as well.

3. They are reasonably accurate only for very low loss probabilities ($< 10^{-6}$). It may not be possible or necessary to meet this requirement. For example, voice traffic is not very demanding in its loss rate requirement (about 10^{-3} according to [23]).

4. The sources studied must be analytically tractable. The traffic model most frequently used is the On/Off bursty source. This approach requires more complex sources to be modeled as a combination of On/Off sources, which may not always be easy to do.

It is important that EB computation schemes capture the non-linear relationship between the total EB and the individual EBs, especially when the statistical multiplexing gain is high. Both [31] and [15] propose admission policies that use a non-linear function of the individual EB to expand the region of application of EB source characterization. The policy proposed by [31] eliminates limitations 1 and 2 above, but still suffers from limitation 3. It also adds complexity by requiring that each call be empirically converted into an equivalent On/Off source feeding a zero-buffer system. The model developed in [15] is simple; however, it overestimates the effective bandwidth of a source considerably, leading to inefficient link utilization.

We consider using regression analysis and adaptive logic network (ALN) modeling approaches to the problem of estimating cell loss rate and delay in the system of figure 2.1. These estimates are then used to compute the effective bandwidth of a call. Our goal is to use methods whose complexity is independent of the number of calls in progress.

The traffic sources being multiplexed in the same queue are likely to have similar peak rates[2]. Therefore, to simplify the problem at hand, we limit ourselves to calls with identical traffic descriptors (homogeneous QoS requirements are common to all of the EB schemes described above).

We use simulations to study the factors affecting cell delay and loss rate. Previously, simulation studies have been aimed at obtaining sets of curves to be used as guidelines in predicting the effective bandwidth of connections. Different sets of curves are stored for calls with widely differing peak rates, even though in other respects (peak to average rate, for example), they may be identical.

We use our simulation results differently. In our first approach, we use multivariate non-linear regression to obtain delay and loss rate as a function of the source characteristics, buffer size, service rate and the number of calls. This equation can be iterated through to obtain the service rate, for a given delay or loss rate requirement, as a function of the other parameters. The time-dependent parameters used in the regression are normalized with respect to the peak rate. Hence, the same regression equation can be used for calls with peak rates very different from that used in the simulation, as long as the other characteristics, such as mean burst length, remain the same. The time to compute the effective bandwidth of a new call is independent of the number of admitted calls because the number of calls is an input parameter to the delay and loss rate models. This approach eliminates the problem in [35] of the large storage space consumed by bandwidth requirement tables.

In our second approach, we train an ALN to predict the delay or cell loss. The ALN can then be inverted to predict the effective bandwidth instead. As before, EB predictions can be made for calls with peak rates very different from that used in the simulation experiments.

Both approaches are shown to predict EB fairly accurately for a wide range of buffer size (1–20 times the burst length), cell loss rate (0.1–10^{-6}) and delay (1–5000 cell slots). This is the non-linear region of interest since analytical methods ([4, 15]) are inaccurate at high cell loss rates.

---

[2] When it is required to multiplex sources with widely differing peak rates and/or QoS requirements, round-robin scheduling among multiple queues is preferable, as it permits fair sharing of resources.

We compare the results obtained by the regression and ALN models with that obtained by the AMS model. Since the AMS model only predicts the CLR (or rather, the overflow probability), we have extended the model to predict the delay as well, and we refer to this as the D-AMS model.

The rest of the chapter is organized as follows. In section 2.3, we describe the simulation experiments that are the basis for the regression and ALN approaches. We define the traffic source model in section 2.3.1. In sections 2.4, 2.5 and 2.6, we describe the ALN, regression and AMS approaches to delay and cell loss estimation, respectively. Section 2.7 contains a comparison of the three models with the (exact) simulation results. In section 2.8, we apply the methods to the computation of effective bandwidth. Our findings are summarized in the concluding section 2.9.

2.2 Notation

The following symbols are used in this chapter:

ALN	Adaptive Logic Network—a variant of an artificial neural network (section 2.4).
D-ALN	The ALN trained to predict cell delay at the switch.
C-ALN	The ALN trained to predict the cell loss rate at the switch.
D-AMS	The AMS model extended to predict cell delay at the switch.
C-AMS	The AMS model that predicts the cell loss rate at the switch.
D-REG	The regression model for cell delay at the switch.
C-REG	The regression model for cell loss rate at the switch.
Clr	The cell loss probability at the switch.
Del	The mean queuing delay (excluding processing delay) at the switch.
t_{ON}	The mean duration of the *On* state of an On/Off source.
t_{OFF}	The mean duration of the *Off* state of an On/Off source.

Bl	The mean number of cells generated by the On/Off source when in the *On* state.
Pcr	The peak cell rate of a single call.
Scr	The average cell rate of a single call.
S	The number of cell buffers dedicated to a single queue of the output port of the switch.
N	The number of calls being multiplexed at the queue under study.
L	The link rate at the output port shared by the various queues including the one under study.
Bw	The cell service rate at the queue under consideration. The portion of the link rate $L - Bw$ is assumed to be fully utilized by the other queues at the output port.
Eb	The effective bandwidth per call; it is the service rate Bw—required to achieve a given QoS—normalized w.r.to Pcr and N.
S/Bl	The ratio of buffer size S to the mean burst length Bl. The delay and cell loss probability are assumed to depend on this ratio rather than on S and Bl individually. This has been verified by simulations for a few cases.
Av/Pk	The ratio of the mean cell rate (Scr) to the peak cell rate (Pcr).

2.3 Experimental setup

2.3.1 Traffic model

Each traffic source has an *On* and an *Off* state. When in the *On* state, the source generates cells at a deterministic rate of Pcr cells/sec; no cells are generated during the *Off* period. The *On* and *Off* periods are exponentially distributed with means t_{ON} and t_{OFF}, respectively (figure 2.2). The N sources are independent, but have identical mean *On* and *Off* periods. The mean burst length, Bl, is given by $Pcr \cdot t_{ON}$, and the average-to-peak-rate ratio, Av/Pk, is given by $\frac{t_{ON}}{t_{ON}+t_{OFF}}$.

2.3.2 Factors affecting cell delay and loss rate

Assume that the traffic from a collection of N identical calls is fed into a finite-sized queue and serviced by a single server (figure 2.1). Then the average cell delay Del and loss rate Clr are influenced by

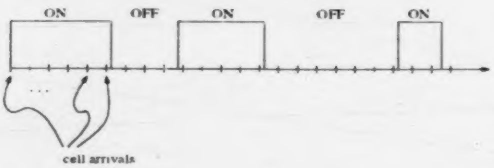


Figure 2.2: Source model

- The characteristics of a single call: mean burst length (Bl), average-to-peak-rate ratio (Av/Pk)[3]. In [9], it is mentioned that the effective bandwidth—for sources whose peak rate is greater than roughly one-tenth of the link bandwidth—is influenced by the burst length. In this study, the peak rate of a source is assumed to be at least ten times smaller than the link rate. This eliminates the need to consider Bl when obtaining an inverted model for Eb. Thus, the peak rate only affects the time-scale of operation without changing the relative effect of the other parameters.

- System parameters: buffer size S, service rate Bw. As mentioned in section 2.2, the delay and loss depend on S and Bl only through the ratio S/Bl. Since our experiments (section 2.3.2) have been performed for a single value of Bl, we can use S and S/Bl interchangeably. Eb is the service rate Bw normalized w.r.t. Pcr and N. Its value lies between Av/Pk and 1.

- Number of calls, N, being multiplexed at the queue.

The reason that we choose Bl and Av/Pk as the parameters representing the characteristics of a single call—rather than using t_{ON} and t_{OFF} directly—is the following. Suppose that we obtain via a simulation study a model for Eb in terms of Clr, t_{ON}, t_{OFF}, S and N. If we need to find the service rate for another source with the same characteristics, but a different peak rate, we will have to scale t_{ON} and t_{OFF} in order to maintain the source characteristics at their original values. The model is no longer applicable since it was obtained using a call with different t_{ON} and t_{OFF}. Therefore, the model must be recalculated for the new connection, which is costly. Our approach avoids this extra computation. By choosing Bl and Av/Pk as the independent variables, their values will remain unchanged when the peak rate is scaled. In that case, we can use the model without modification.

---

[3] The coefficient of variation of inter-cell time may be a more accurate characterization of the source than the average-to-peak ratio. However, the latter, in conjunction with the burst length, does uniquely characterize the source. This, as pointed out in [36], is not the case when Av/Pk is used as the sole characteristic of the source.

Of the five factors mentioned previously, we chose three factors for a full-factorial study—S, Eb, N. The traffic generated by a single source can be viewed as a video session, generated according to the model described in [16], with $t_{ON} = 25$ ms (representing the mean t_{ON} duration of the model traffic), $t_{OFF} = 35$ ms and $Pcr = 14150$ cells/s. This results in $Bl = 353$ cells and $Av/Pk = 0.417$.

One might object that an Av/Pk of 0.417 and a mean burst length of 353 cells represent a rather well-behaved source. Decreasing Av/Pk to, say, 0.1 would necessitate considering normalized effective bandwidth from 0.1 to 0.9, instead of 0.42 to 0.9 as is done now (section 2.3.3). To predict the effective bandwidth with the same accuracy would require doubling the number of simulation experiments. Increasing Bl to, say, 1000 would only mean that the lower limit of the buffer size would be 1000 rather than the current value of 353. Since the goal of this study is to demonstrate the utility of regression and neural networks in bandwidth prediction (a non-trivial problem even assuming identical and well-behaved On/Off sources), the exact characteristics of the source are assumed to be of secondary importance.

2.3.3 Training set data

The levels chosen for the three factors mentioned before are shown in table 2.1. When the number of calls is large, linear approximations are expected to be accurate (figure 2.3 left) and hence we do not consider more than 25 calls.

The same observation applies when the buffer size is much larger than Bl (figure 2.3 right) and so we only consider a maximum buffer size of 5000 (this value was

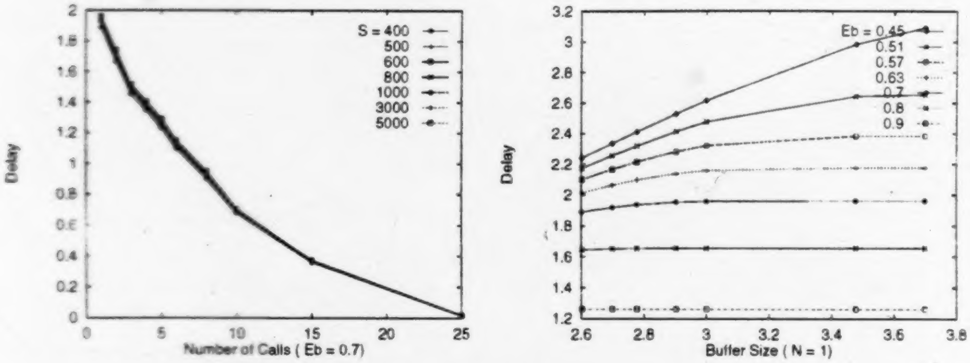


Figure 2.3: Delay vs. N (left) Delay vs. S (right)

Factor	*Levels*	*Number of levels*
Number of calls	1–5, 6, 8, 10, 15, 25	10
Buffer size	400, 500, 600, 800, 1000, 3000, 5000	7
Service rate	0.45–0.67 in steps of 0.02, 0.67–0.9 in steps of 0.03	19

Table 2.1: Levels of factors for training set

obtained from our initial experiments). Since *Bl* is 353 cells, the maximum value of S/Bl is 22. As we are mainly interested in predicting *Eb*—recall that its value lies in $[Av/Pk, 1]$—we used a fine granularity of 0.02. We found that when *Eb* was close to its minimum value of 0.417 (corresponding to Av/Pk), the steady-state values of *Clr* and *Del* were quite difficult to obtain accurately, even with very long simulations. Therefore, we opted to use a minimum *Eb* of 0.45.

In each experiment, 150 million cells were simulated since we were interested in cell loss rates of the order of 10^{-6}. A total of $10 \cdot 7 \cdot 19 = 1330$ experiments were run using SMURPH [13], a simulation tool developed at the University of Alberta. This tool allowed us to distribute experiments over roughly seventy machines on the local departmental network, utilizing idle cycles. The experiments were repeated with a different seed for the random number generator to improve the reliability of the cell delay and loss rate estimates.

2.3.4 Test set data

The levels chosen for the three factors above are shown in table 2.2. The values of Bl and Av/Pk were the same as in the training set. However, the peak cell rate (Pcr) was chosen to be 1000 cells/sec. This was done so that it can be verified that the regression and ALN models could correctly predict the cell delay and loss rate when the peak rate was different from that used in the training set. The mean on and off times were suitably altered to maintain Bl and Av/Pk at their values in the training set. The levels for the factors in the test set have been chosen to test the developed models on both interpolation and extrapolation. As in the case of the training set, several million cells were simulated in each experiment. Experiments were repeated to obtain better estimates.

Factor	*Levels*	*Number of levels*
Number of calls	1, 2, 3, 7, 9, 12, 20, 30	8
Buffer size	370, 550, 900, 2000, 4000, 10000.	6
Service rate	0.44, 0.48, 0.52, 0.56, 0.6, 0.65, 0.7, 0.75, 0.8 0.85, 0.9, 0.95	12

Table 2.2: Levels of factors for test set

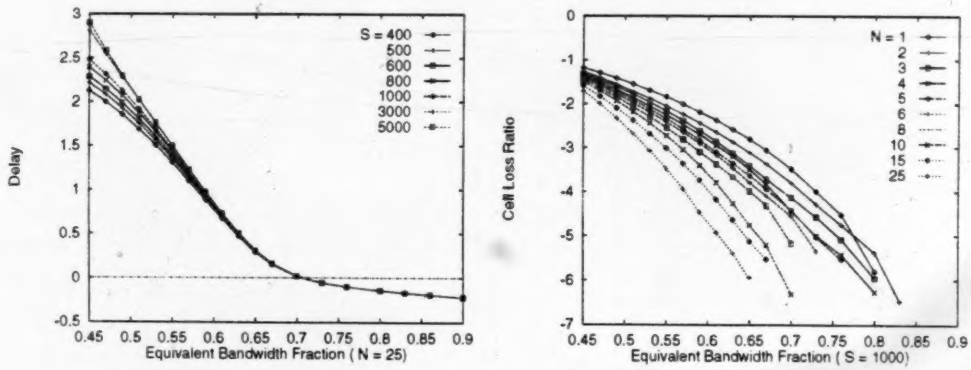


Figure 2.4: Delay vs. Eb (left) Loss rate vs. Eb (right)

Figure 2.4 (left) shows the variation of cell delay with Eb for varying N and $S = 1000$ while figure 2.4 (right) shows the loss rate as a function of Eb. The qualitative information in these figures is used in deriving the regression and ALN models.

2.4 Adaptive logic networks

An adaptive logic network (ALN) is a type of artificial neural network ([6]). One way of describing ALNs is as a layer of perceptrons whose outputs are processed by layers of AND and OR logic gates [5]. In that form, the ALN maps vectors of real values in Euclidean n-dimensional space to boolean values. The first layer of computing units in the network consists of linear threshold units (perceptrons) that output 1 if an inequality of the form $w_0 + w_1 \times x_1 + w_2 \times x_2 + \ldots + w_n \times x_n >= 0$ is satisfied, otherwise they output 0 . The coefficients w_i of the expression are called the weights of the unit. The boolean outputs of the first layer of units are combined by a tree expression of AND and OR operators of arbitrary fan-in to produce the output of the ALN. An ALN can be viewed as a function $\Re^n \rightarrow \Re$ that produces a 1 as output on or under the graph of the function when given the $n-1$ inputs $x_1, \ldots, x_{n-1}$ and the output x_n that is to be learned.

It is also possible to view the ALN in terms of the real-valued function it represents. A functional computation can be derived from the ALN by taking combinations of linear functions where the combining operators are maximum and minimum of functions. If x_n is the output variable of the ALN, the weights of the first layer of units are normalized to have $w_n = -1$ and the inequality of a unit is turned into a function of the form $x_n = w_0 + w_1 \times x_1 + w_2 \times x_2 + \ldots + w_{n-1} \times x_{n-1}$.

The tree of maximum and minimum operators has the same form as the tree of OR and AND operators respectively. The linear functions have weights which are adapted based on training data consisting of vectors $x_1, \ldots, x_n$ that represent the function graph. The algorithm is like least squares fitting of linear pieces to data points, where a linear piece is only active for a subset of the training points. Given $x_1, \ldots, x_{n-1}$, a subtree of a node contains the active linear piece if its value (computed using the maximum and minimum operators according to the subtree) is less (or, respectively, greater) than the value of any other subtree for an AND- (or, respectively, OR-) node.

The software that was used in the experiments refines the piecewise linear functions produced by the above ALN by inserting quadratic *fillets* at each junction of two linear pieces so that the overall function is continuously differentiable.

For ATM traffic characterization, there are several advantages of ALNs over other types of neural networks:

- The normalized weights of linear pieces that are active are the values of the partial derivatives of the output variable with respect to the input variables. The weights can be directly constrained to lie within prescribed intervals. The partial derivatives of the fillets lie between those of the linear pieces being joined. Hence the partial derivatives of the learned functions can be directly controlled. For example, forcing a weight to be positive forces the learned function to be strongly monotonic increasing in the corresponding variable.

- If an ALN represents a function $x_n = \mathrm{f}(x_1, \ldots, x_{n-1})$ which is monotonic increasing in some variable x_i, then a functional computation can be derived from that ALN: $x_i = g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ which computes the corresponding function inverse. This uses the coefficients of the same linear pieces, combined in a different way.

- The ALN does not require the predictor variables to be scaled or normalized. This speeds up the model development process as well the use of the model for prediction.

Note that most performance functions are naturally monotonic. Forcing the learned function to be monotonic or convex makes it difficult for the function learned by an ALN to be influenced by the noise in training points; hence over-training which prevents good generalization in other neural networks may be avoided in some cases. The use of these advantageous properties is demonstrated in sections 2.4.2 and 2.4.3.

2.4.1 Choosing the epsilon values

The ALN software used allows the user to specify a smoothing parameter associated with each variable that expresses the half-length of an interval which has to be covered by each point in a training set in each axis. For example if a real value is sampled every 0.8 units, then epsilon might be set to 0.4, half the distance between neighboring points. Increasing epsilon has the effect of smoothing the function in the direction of the variable, but too large an epsilon makes it impossible for the ALN to discriminate between points separated by distance epsilon. In the above example, a value for epsilon greater than 0.8 would make the ALN unable to distinguish between neighboring points.

In the case of the effective bandwidth problem, we have three input variables: Eb, $log(S)$[4] and N. The minimum value of the epsilon, $\epsilon_{i,min}$, for an input variable, i, is given by half the smallest interval, $I_{i,min}$, between two adjacent levels of that variable in the training set. The maximum value for epsilon, $\epsilon_{i,max}$, is given by $I_{i,min}$. As mentioned before, this prevents over-smoothing of the learned function. In order to prevent over-training, it is customary to evaluate at periodic intervals the trained ALN on the test set. When a point is reached at which the error on the test set increases in spite of a decrease in the error on the training set, training can be halted. This has the disadvantage that the final evaluation on the test set does not really test the generalization of the ALN, since the ALN has "seen" the test set.

Predictor Variable	**Smallest interval**	$\epsilon_{i,min}$	$\epsilon_{i,max}$
Eb	0.02	0.01	0.02
$log(S)$	0.1	0.05	0.1
N	1	0.5	1

Table 2.3: Epsilons for D-ALN and C-ALN

---

[4] Because of the large range of buffer size being considered, we use instead the logarithm of the buffer size.

Instead, we divide the training set into two portions. Set 1 contains the simulation data for $Eb = 0.45, 0.49, 0.53, \ldots$ and is used as the training data. Set 2 contains the data for $Eb = 0.47, 0.51, 0.55, \ldots$. After every every set of ten passes (epochs) through the training data (set 1), the trained ALN is evaluated on set 2. The test data is used only when a satisfactorily trained ALN is obtained. It must be noted that this partitioning of the training data has the effect of changing ϵ_{min} and ϵ_{max} for Eb to 0.02 and 0.04, respectively.

Over-training is also reduced by constraining the slope of Del w.r.t. the other variables; in this case, delay was constrained to *decrease* monotonically as a function of Eb and N, and *increase* monotonically as a function of buffer size.

2.4.2 ALN model for cell delay

The inputs to the D-ALN are S, Eb, N and Del. Because of the wide ranges of S and Del, we chose to train the D-ALN on $log(S)$ and $log(Del)$, respectively. Since the ALN is scale-invariant, we did not have to normalize the inputs as is usually done with other neural networks.

The software allows the user to specify an epsilon for the output variable, ϵ_{op}, as well. Whenever the root mean square error (RMSE) on the training set is greater than ϵ_{op}, the ALN automatically grows in size to reduce the error. If ϵ_{op} is set to an unnecessarily small value, the resulting ALN may become very large without corresponding reduction in error on the test set. Too large a value of epsilon may prevent the ALN from learning satisfactorily due to inadequate size.

In order to determine the optimum epsilon on the output variable, ϵ_{del}, for the D-ALN, three different values were tried. Table 2.4 shows the RMSE on the training set and the average relative error (ARE) on set 2 for each value of epsilon. From the table, we can see that the lowest ARE (0.73 %) is is achieved for $\epsilon = 0.01$. Hence ϵ_{del} is chosen as 0.01. The table also shows the point at which training, if continued, would result in poor generalization. For example, when $\epsilon = 0.01$, the ARE decreases

Epoch	$\epsilon = 0.05$	$\epsilon = 0.01$	$\epsilon = 0.003$
10	<0.0273, 1.28 %>	<0.0271, 1.16 %>	<0.0271, 1.16 %>
20	<0.0162, 0.84 %>	<0.0142, 0.8 %>	<0.0142, 0.8 %>
30	<0.0139, 0.79 %>	<0.0115, 0.79 %>	<0.0115, 0.78 %>
40	<0.0128, 0.79 %>	<0.0096, 0.77 %>	<0.0095, 0.75%>
50	<0.0112, 0.81 %>	<0.0084, 0.75 %>	<0.0085, 0.75 %>
60	<0.0114, 0.88 %>	<0.0079, 0.76 %>	<0.0079, 0.79 %>
70	<0.0108, 0.78 %>	<0.0070, 0.73 %>	<0.0070, 0.78 %>
80	<0.0110, 0.97 %>	<0.0070, 0.8 %>	<0.0068, 0.84 %>
90	<0.0112, 0.97 %>	<0.0064, 0.81 %>	<0.0064, 0.87 %>
100	<0.0106, 1.05 %>	<0.0058, 0.79 %>	<0.0060, 0.84 %>

Table 2.4: Choosing the output epsilon for D-ALN

continuously for the first seventy epochs and then increases. This indicates that training should be stopped after seventy epochs.

Next, we try to determine the optimum epsilon for each of the input variables in a similar manner. For each variable, the values tried are ϵ_{min}, ϵ_{max} and $\frac{\epsilon_{min}+\epsilon_{max}}{2}$. The epsilon resulting in the lowest ARE is then chosen.

The software tries to lower the RMSE to a specified level, rms_{min}. Setting rms_{min} to a value much below the ϵ_{op} has little advantage. On the other hand, setting rms_{min} to ϵ_{op} may not result in a satisfactorily trained ALN. Table 2.5 shows the RMSE and ARE for three different values of rms_{min}. It is clear that there is little advantage in choosing $rms_{min} < 0.01$. It can also be seen that training should be stopped after 30 epochs.

Figure 2.5 compares the estimates of the D-ALN on the training and test sets with simulation results. Since the test set was generated for Pcr = 1000 cells/sec while the training set used Pcr = 14150 cells/sec, the figure shows that the trained ALN can be used to predict delays for other values of Pcr fairly accurately—as long as Bl and Av/Pk remain the same. Section 2.7.1 presents a comparison of the D-ALN model with the D-AMS and D-REG models.

Epoch	$rms_{min} = 0.01$	$rms_{min} = 0.005$	$rms_{min} = 0.001$
10	<0.0277, 1.2 %>	<0.0277, 1.2 %>	<0.0277, 1.2 %>
20	<0.0143, 0.71 %>	<0.0143, 0.71 %>	<0.0143, 0.71 %>
30	<0.0112, 0.69 %>	<0.0112, 0.69 %>	<0.0112, 0.69 %>
40	<0.0103, 0.71 %>	<0.0103, 0.71 %>	<0.0103, 0.71 %>
50	<0.0086, 0.72 %>	<0.0086, 0.72 %>	<0.0086, 0.72 %>
60	-	<0.0076, 0.77 %>	<0.0076, 0.77 %>
70	-	<0.0070, 0.84 %>	<0.0070, 0.84 %>
80	-	<0.0068, 0.85 %>	<0.0068, 0.85 %>
90	-	<0.0064, 0.82 %>	<0.0064, 0.82 %>
100	-	<0.0063, 0.90 %>	<0.0063, 0.9 %>

Table 2.5: Choosing rms_{min} for D-ALN

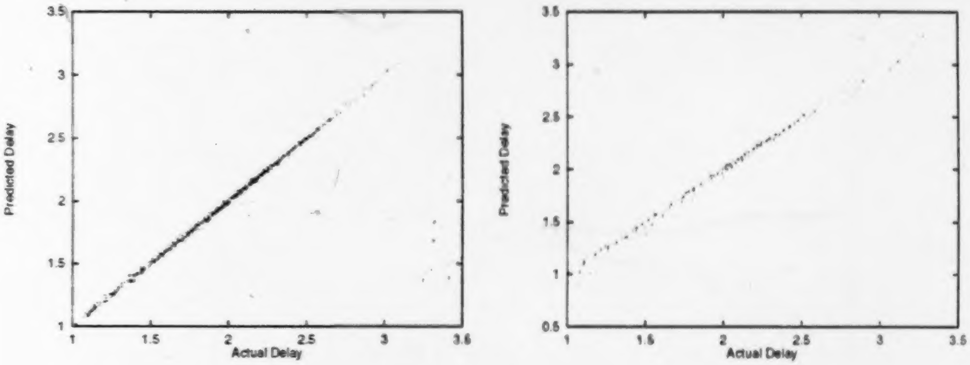


Figure 2.5: D-ALN model predictions: training set (left) test set (right)

2.4.3 ALN model for cell loss rate

The inputs to the C-ALN are the same as those to the D-ALN. The ALN is trained on $log(Clr)$. The points in the training set for which $Clr = 0$ are eliminated. This results in the C-ALN being trained on 899 data points, which is still adequate considering that we have only three input variables.

As in the case of the D-ALN, training was done on one half of the training set, with the other half being used to check for over-training. The learned function was constrained by specifying that the Clr is a monotonic decreasing function of Eb, N and S.

Optimum epsilons for Eb, $log(S)$ and N were obtained, in each case being equal to the respective minimum epsilons. The optimum output epsilon, ϵ_{clr}, was found to be 0.05. The best setting for rms_{min} was also 0.05. Training was stopped after 40 epochs.

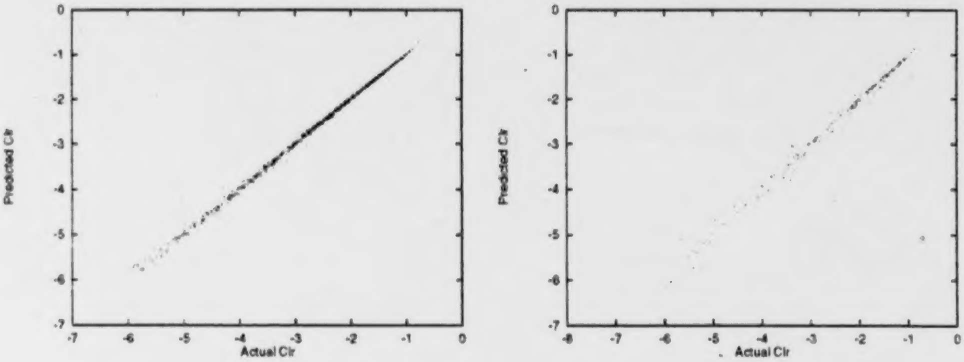


Figure 2.6: C-ALN model predictions: training set (left) test set (right)

Figure 2.6 compares the predictions of the C-ALN model on the training and test sets with simulation results. As in the case of cell delay, the results obtained by evaluating the trained model on the test set show that the ALN can be used to predict loss rates for different values of Pcr. Section 2.7 compares the loss rate predictions by the C-AMS, C-REG and C-ALN models with the (exact) simulation results.

The performance of the C-ALN model on the test set is not as good as the D-ALN model. The maximum relative error on the test set (19.9%) is almost twice that of the D-ALN (11.8%). We can also see a larger scatter at low loss ($< 10^{-6}$) in figure 2.6. This is primarily because of the smaller training set due to the missing elements corresponding to zero loss rate. These values could be replaced by more accurate estimates from longer simulation runs at high service rates and large buffer sizes.

Figure 2.7 (left) compares D-ALN model predictions for $N = 30$—a value not in the training set—with simulation values. Figure 2.7 (right) shows the ability of the C-ALN model to extrapolate accurately.

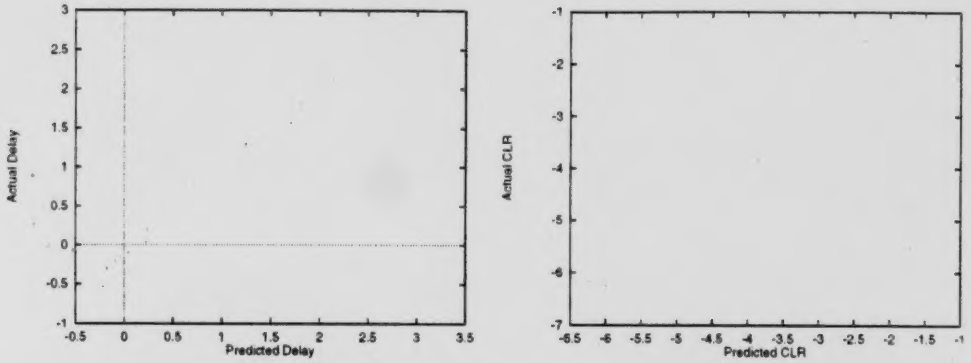


Figure 2.7: Extrapolation for N=30: D-ALN model (left) C-ALN model (right)

2.4.4 Residual analysis for ALN models

Since the ALN uses the ordinary least squares (OLS) principle to determine the orientation of the linear pieces, the trained ALN is subject to the assumptions inherent in the use of OLS.

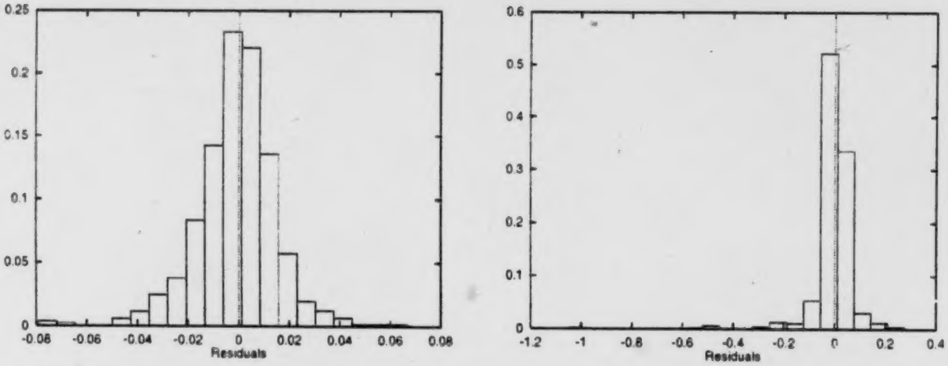


Figure 2.8: Histogram of residuals: D-ALN model (left) C-ALN model (right)

The assumptions made in the case of OLS are [32]

- Predictor variables are non-stochastic and measured without error. Since the predictor variables are controlled inputs to the simulation, this statement is true.

- Model error terms follow a normal probability distribution. This can be seen to be approximately true from the histogram plots of $log(Del)$ and $log(Clr)$ residuals (figure 2.8).

- Any two errors are independent of each other. The presence of correlation reduces the reliability of the model. To verify this, we plot the residuals against the values obtained by simulation (figure 2.9).

- Model error terms have zero means, are uncorrelated, and have constant variances. The first of these can be seen to be true by observing that the standard deviation limits appear equidistant from the horizontal axis (figure 2.9). The second condition is generally true for databases compiled from controlled laboratory experiments. The third assumption is discussed below.

From the plot of **D-ALN** residuals, we can see that the errors are randomly distributed on either side of the horizontal axis, indicating that there is no systematic error. Since the log function is nonlinear for values of the abscissa < 10, we have eliminated these values from the plot. In any case, we are more interested in delay predictions at higher delays.

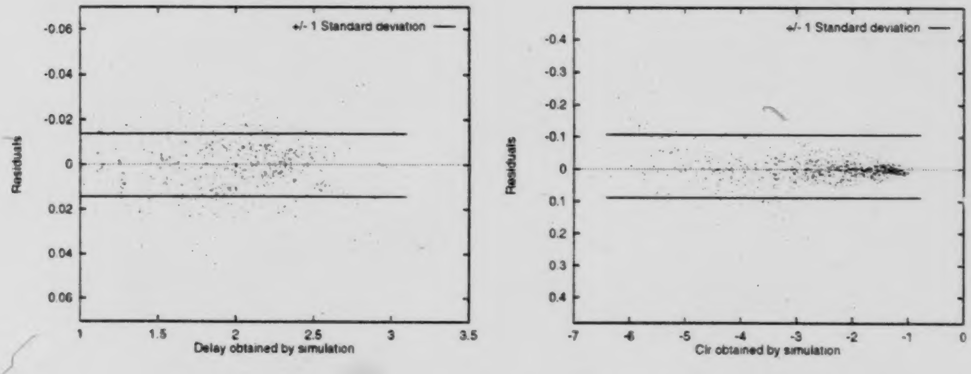


Figure 2.9: Plot of residuals: **D-ALN** (left) **C-ALN** (right)

The plot of **C-ALN** residuals shows that the residuals are fairly random for $Clr > 10^{-6}$. For lower cell loss, the simulation results themselves are inaccurate. The residuals are seen to increase in magnitude for $Clr < 10^{-4}$, leading us to conclude that there is a small amount of heteroscedasity (unequal variances).

2.5 Regression analysis

Regression analysis is considered indispensable as a data analysis technique in a variety of disciplines. So far, however, the models developed for bandwidth characterization in ATM networks have been analytical models developed from basic principles. Given the variety of traffic expected on ATM networks as well as the complexity of this traffic, it will not always be possible to develop such models. Either the development of an analytical model may be time-consuming or the assumptions needed to make the problem tractable may be too limiting.

Moreover, even if a model is obtained by analytical means, it is often too complex to be used for real-time purposes. We feel that a regression model based on simulation data can be developed in a relatively shorter time and yield a simple relationship between the response and predictor variables. This simple relation can be utilized for prediction of the response variable for new values of the predictor variables. In our study, for example, we wish to predict cell delay and loss rate for values of Eb, S/Bl and N not contained in the simulation data. In addition, we can iterate through the relationships for Dcl and Clr to find the Eb necessary to satisfy a given QoS. Because of the simple relationship between the response and predictor variables, the effective bandwidth can be found rapidly. As a result, the regression models are useful for real-time call admission control (CAC).

The data upon which the regression analysis is based must be as representative as possible to ensure that inferences or predictions made are accurate. A common misuse of regression methodology is extrapolation—attempting to predict the response variable for values of predictor variables not represented in the database. To reduce the temptation for this, we have tried to make our database as large as is necessary to ensure that most meaningful queries can be answered by the model.

The domain of the predictor variables are as follows:

$log(S)$ S can range from 400 to 5000 cells. Preliminary studies showed that, for the traffic under consideration, the delay and cell loss rate values did not change significantly when the buffer size was increased further.

Eb It ranges from 0.45 to 0.9, since the effective bandwidth is normalized w.r.to the peak rate.

N The number of calls ranges from 1 to 25. When the number of calls is large, it is expected that the law of large numbers will apply; therefore the change in delay or loss rate, as the number of calls increases further, will be small.

2.5.1 Regression model for delay

In developing the regression model, it is necessary to ensure that the model does not fit the noise in the training data. The coefficient of determination (R^2) is not a good indicator of the goodness of the model as it can be artificially inflated—by adding parameters to the model—and made arbitrarily close to 1.

Therefore, we need to balance the increased accuracy due to smaller residuals against the cost of a more complex model. The CP [20] criterion is widely used for this purpose. It is defined as

$$CP = \frac{sse}{mse\_p_{max}} - (n - 2p) \tag{2.2}$$

where

- sse is the sum of squares of residuals with p parameters
- mse_p_{max} is the mean sum of squares of residuals for a model that includes as many parameters $p = p_{max}$ as possible; in a sense, the best model possible from the point of view of the smallest residuals. The mean sum of squares of residuals

is computed as $mse\_p_{max} = sse\_p_{max}/(n - p_{max})$, where $sse = sse\_p_{max}$ for the model with p_{max} parameters.

- n is the number of training data points.

In order to select the "best" model, we look for models whose CP value is reasonably close to the number of parameters p in the model. The model with the smallest number of parameters is then chosen from this subset of models. Models with a considerable lack of fit will usually have CP values much greater than p. It is possible that a model with fewer parameters may have a smaller CP, but that the larger CP of a model with more parameters is closer to its p (indicating lower bias). In such cases, the choice of a model is largely a matter of judgment.

We use the well-known statistical package SPSS [33]. Because of the wide range of cell delay, we choose $log(Del)$ as the response variable. Based on the simulation plots, the following observations can be made:

- $log(Del)$ decreases almost exponentially as a function of Eb (figure 2.4 left).
- The variation of $log(Del)$ as a function of S/Bl has the form

  $$a_{\infty}(Eb) \times (1 - e^{-\beta(Eb) \times S/Bl})$$

  where $a_{\infty}(Eb)$—a function of Eb alone—is the value of $log(Del)$ when S/Bl is very large (figure 2.3 right).
- $log(Del)$ decreases exponentially as a function of N (figure 2.3 (left)). The form of the function is

  $$c(1) \times e^{-\gamma(Eb) \times (N-1)}$$

  where coefficient $c(1)$—a function of S/Bl and Eb—is the value of $log(Del)$ when $N = 1$.

We do not include S/Bl in the function for N or vice-versa because of an analysis which determined that there is very little interaction between N and S/Bl. Selecting

$N = 1$ as the origin for the number of calls, we use $N - 1$ instead of N in the model. The composite model is of the form:

$$log(Del) = f(Eb) \times (1 - e^{-\beta(Eb) \times S/Bl}) \times (e^{-\gamma(Eb) \times (N-1)}) \qquad (2.3)$$

The D-REG model is obtained in three stages. In the first stage, we set $N = 1$ and buffers $S = 5000$ (to ensure a large value of S/Bl). The second and third terms in equation 2.3 can be seen to reduce to 1. We can now obtain a sub-model for $log(Del)$ in terms of $f(Eb)$.

Table 2.6 shows the value of CP for several different forms of $f(Eb)$, the first term in equation 2.3. The model with seven parameters has the smallest residuals and hence it is selected as the baseline for comparing other models. From equation 2.2, the CP for this model is closest to $p = p_{max} = 7$ because the first term in the equation reduces to $n - p_{max}$. It can be seen that all models except the last exhibit considerable bias; that is, they lie far away from the $CP = p$ line. Hence we choose the last model for $f(Eb)$. In the second stage, we choose $N = 1$ as before, but place no restriction on S. We have obtained the first term in equation 2.3 and the third term reduces to 1. We perform a regression analysis to obtain the values of the parameters in the sub-model $1 - e^{-\beta(Eb) \times S/Bl}$. Table 2.7 shows CP values for the different sub-models

Model	*Number of Parameters (p)*	*CP*
$a \times e^{(-b \times Eb + c)}$	3	996.483
$a \times e^{(-b \times Eb + c)} + d$	4	997.41
$a \times Eb \times e^{(-b \times Eb + c)} + d$	4	1332.08
$a \times e^{(-b \times Eb + c)} \times (d - e \times Eb)$	5	1000.48
$a \times e^{(-b \times Eb + c)} \times (d - e \times Eb) + f$	6	1001.41
$a \times e^{(-b \times Eb + c)} \times (d \times bw - e \times Eb^2) + f$	6	1682.71
$a \times e^{(-b \times Eb + c)} \times (d - e \times Eb^2) + f$	6	877.58
$a \times e^{(-b \times Eb + c)} \times (d - e \times Eb + f \times Eb^2)$	6	84.047
$a \times e^{(-b \times Eb + c)} \times (d - e \times Eb + f \times Eb^2) + g$	7	7

Table 2.6: CP criterion for D-REG sub-model (stage 1)

tried in the second stage. Each of the sub-models in this stage is multiplied by the sub-model chosen in the first stage to obtain the predicted values of $log(Del)$. We see that only the sub-models with $CP = 8.42$ (3 parameters) and $CP = 5$ (5 parameters) are adequate. The former has lower CP, but the CP of the latter model is closer to its p, indicating smaller bias. We pick the smaller sub-model in this case, opting for simplicity rather than additional accuracy.

Our choice is supported by the fact that the values of g and h are close to 1 (1.0016 and 1.15, respectively). This indicates that while the four parameter sub-model tracks the simulation data (and noise) better, the sub-model with three parameters is probably more representative of the real system. In the third stage, we place no restriction on N. Table 2.8 CP values for different forms of $\epsilon^{-\gamma(Eb)\times(N-1)}$, the last product term in equation 2.3. Interestingly enough, although the single parameter sub-model using $ln(N)$ has a lower CP than that using $N-1$, this is not true in the case of the two parameter sub-models.

Model	*Number of Parameters (p)*	*CP*
$(1-\epsilon^{(-i\times S/Bl)}$	1	27824.9
$(1-\epsilon^{((i-j\times Eb)\times S/Bl)}$	2	11335.3
$(g-h\times\epsilon^{(-i\times S/Bl)}$	3	23673.6
$(1-\epsilon^{((i-j\times Eb)\times S/Bl-k\times Eb)}$	3	8.42016
$(g-h\times\epsilon^{(-i\times S/Bl-j\times Eb)}$	4	509.434
$(g-h\times\epsilon^{((i-j\times Eb)\times S/Bl)}$	4	60.5071
$(g-h\times\epsilon^{((i-j\times Eb)\times S/Bl-k\times Eb)}$	5	5

Table 2.7: CP criterion for D-REG sub-model (stage 2)

Each of the sub-models in this stage is multiplied by the sub-model chosen in the first and second stages to obtain the predicted values of $log(Del)$. From table 2.8, we see that only the sub-models with $CP = 4.84$ (4 parameters) and $CP = 5$ (5 parameters) lie reasonably close to the $CP = p$ line. We choose the smaller sub-model with $p = 4$.

Model	*Number of Parameters (p)*	*CP*
$e^{(-m\times(N-1))}$	1	67319.2
$e^{(-m\times ln(N))}$	1	67191.2
$e^{((m-n\times Eb)\times(N-1))}$	2	7840.99
$e^{((m-n\times Eb)\times ln(N))}$	2	10996.7
$e^{((m-n\times Eb+o\times Eb^2)\times(N-1))}$	3	258.039
$e^{((m-n\times Eb+o\times Eb^2+p\times Eb^3)\times(N-1))}$	4	4.83898
$e^{((m-n\times Eb+o\times Eb^2+p\times Eb^3+q\times Eb^4)\times(N-1))}$	5	5

Table 2.8: *CP* criterion for D-REG sub-model (stage 3)

The final model for $log(Del)$ has the form:

$$log(Del) = f(Eb) \times g(S/Bl) \times h(N-1) \tag{2.4}$$

where

$$f(Eb) = a \times e^{-b\times Eb+c} \times (d - e \times Eb + f \times Eb^2) + g$$

$$g(S/Bl) = (1 - e^{(i-j\times Eb)\times S/Bl-k\times Eb})$$

$$h(N\text{-}1) = e^{(m-n\times Eb+o\times Eb^2+p\times Eb^3)\times(N-1)}$$

We now examine the performance of the delay regression model. Figure 2.10 compares values predicted by the D-REG model from the training and test sets against values obtained by simulation. Since the test set was generated for Pcr = 1000 cells/sec while the training set used Pcr = 14150 cells/sec, this shows that the model can be used to predict delays for new values of Pcr as long as Bl and Av/Pk remain the same (section 2.3.4). Section 2.7.1 compares the predictions of the D-REG model with those of the D-ALN and D-AMS models.

2.5.2 Regression model for cell loss rate

In developing the regression model for cell loss rate, we follow the same procedure as for the delay model. The *CP* criterion described in section 2.5.1 is used to prevent over-

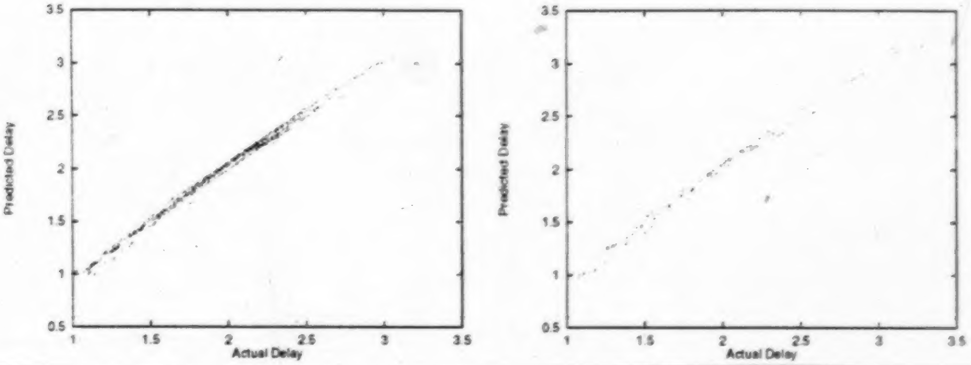


Figure 2.10: D-REG model predictions: training set (left) test set (right)

fitting. Because of the wide range of cell loss rate, we choose *log*(Clr) as the response variable for the model. Based on the simulation plots, the following observations can be made.

- *log*(Clr) decreases almost linearly as a function of *Eb* since the loss rate appears to be an exponential function of *Eb* (figure 2.4 right).

- The form of the variation of *log*(Clr) with S is

  $$a_\infty(Eb) \times (e^{-b(Eb)\times S/Bl})$$

  where a_∞—a function of *Eb* alone—is the value of *log*(Clr) when S/Bl is large.

- *log*(Clr) decreases exponentially as a function of N. The form of the function is

  $$c(1) \times e^{-\gamma(Eb,S/Bl,N-1)}$$

  where coefficient $c(1)$—a function of S/Bl and *Eb*—is the value of *log*(Clr) when $N = 1$.

We do not include N in the function for S/Bl for the same reasons as in the case of the D-REG model. We select the smallest value of S (400) in the training set to correspond to the origin for S/Bl and hence replace S/Bl by $S/Bl - 400/353$.

Similarly, we select $N = 1$ as the origin for the number of calls. The composite model is therefore of the form:

$$log(Clr) = f(Eb) + (e^{-\beta(Eb)\times(S/Bl-400/353)}) + (e^{-\gamma(Eb,S/Bl,N-1)}) \qquad (2.5)$$

The C-REG model is also obtained in three stages. In the first stage, we set $N = 1$ and $S = 400$. The second and third terms in equation 2.5 reduce to 1. We can now model $f(Eb)$ (this function must contain a additive constant since the second and third terms of the above equation reduce to unity when $N = 1$ and $S = 400$). Table 2.9 shows CP values for different forms $f(Eb)$, the first term in equation 2.5. The sub-model with 6 parameters is the only possible choice in this case. In the second stage, we choose $N = 1$ as before, but place no restriction on S. The first term in equation 2.5 has already been determined and the third term reduces to 1. Table 2.10 shows CP values for the different sub-models tried for $\beta(Eb)$. Each of the sub-models in this stage is added to the sub-model chosen in the first stage to obtain the predicted values of $log(Clr)$. From table 2.10, we see that only the sub-model with 4 parameters is adequate. In the third stage, we place no restriction on N. Table 2.11 shows CP values for models of $\gamma(Eb, S/Bl, N-1)$, the last product term in equation 2.5. Each of the sub-models in this stage is added to the sub-model chosen in the first and second stages to obtain the predicted values of $log(Clr)$. From table 2.11, we see that only the sub-model with 7 parameters is sufficiently accurate. The final model for $log(Clr)$ has the form

Model	*Number of Parameters (p)*	*CP*
$a + b \times Eb$	2	15124.8
$a + b \times Eb + c \times Eb^2$	3	2864.83
$a + b \times Eb + c \times Eb^2 + d \times Eb^3$	4	496.346
$a + b \times Eb + c \times Eb^2 + d \times Eb^3 + e \times Eb^4$	5	71.2434
$b \times Eb + c \times Eb^2 + d \times Eb^3 + e \times Eb^4 + f \times Eb^5$	5	35.4301
$a + b \times Eb + c \times Eb^2 + d \times Eb^3 + e \times Eb^4 + f \times Eb^5$	6	6

Table 2.9: CP criterion for C-REG sub-model (stage 1)

Model	*Number of Parameters (p)*	*CP*
$g \times (S/Bl - 400/353)$	1	10503.8
$(g + h \times Eb) \times (S/Bl - 400/353)$	2	727.174
$(g + h \times Eb + i \times Eb^2) \times (S/Bl - 400/353)$	3	44.8915
$(g + h \times Eb + i \times Eb^2 + j \times Eb^3) \times (S/Bl - 400/353)$	4	4

Table 2.10: *CP* criterion for C-REG sub-model (stage 2)

Model	*Number of parameters (p)*	*CP*
$(k + l \times Eb) \times (N - 1)$	2	2248.53
$(k + l \times (S/Bl - 400/353)) \times (N - 1)$	2	12952.0
$(k + l \times Eb + m \times (S/Bl - 400/353)) \times (N - 1)$	3	2248.34
$(k + l \times Eb) \times (N - 1) + m \times (N - 1)^2$	3	306.799
$(k + l \times Eb + m \times (S/Bl - 400/353)) \times (N - 1) + n \times (N - 1)^2$	4	291.195
$(k + l \times Eb) \times (N - 1) + (m + n \times Eb) \times (N - 1)^2$	4	173.233
$(k + l \times Eb) \times (N - 1) + m \times (N - 1)^2 + n \times (N - 1)^3$	4	194.986
$(k + l \times Eb + m \times (S/Bl - 400/353)) \times (N - 1) + (n + o \times Eb) \times (N - 1)^2$	5	976.407
$(k + l \times Eb) \times (N - 1) + (m + n \times Eb) \times (N - 1)^2 + o \times (N - 1)^3$	5	94.8352
$(k + l \times Eb) \times (N - 1) + (m + n \times Eb) \times (N - 1)^2 + (o + p \times Eb) \times (N - 1)^3$	6	31.3509
$(k + l \times Eb + m \times (S/Bl - 400/353)) \times (N - 1) + (n + o \times Eb) \times (N - 1)^2 + p \times (N - 1)^3$	6	69.6996
$(k + l \times Eb + m \times (S/Bl - 400/353)) \times (N - 1) + (n + o \times Eb) \times (N - 1)^2 + (p + q \times Eb) \times (N - 1)^3$	7	7

Table 2.11: *CP* criterion for C-REG sub-model (stage 3)

$$log(Clr) = f(Eb) + g(S/Bl - 400/353) + h(Eb, S/Bl, N - 1) \qquad (2.6)$$

where

$f(Eb) = a + b \times Eb + c \times Eb^2 + d \times Eb^3 + e \times Eb^4 + f \times Eb^5$

$g(S/Bl\text{-}400/353) = (g + h \times Eb + i \times Eb^2 + j \times Eb^3) \times (S/Bl - 400/353)$

$h(Eb,\ S/Bl,\ N\text{-}1) = (k + l \times Eb + m \times (S/Bl - 400/353)) \times (N - 1) + (n + o \times Eb) \times (N - 1)^2 + (p + q \times Eb) \times (N - 1)^3$

We now examine the performance of the C-REG model. Figure 2.11 compares the loss rates predicted by the model with the values obtained by simulation. The results obtained by evaluating the model on the test set show that the model can be used to predict loss rate for new values of *Pcr*. Section 2.7.2 compares the loss rate predictions by the C-AMS, C-ALN and C-REG models against the (exact) simulation results.

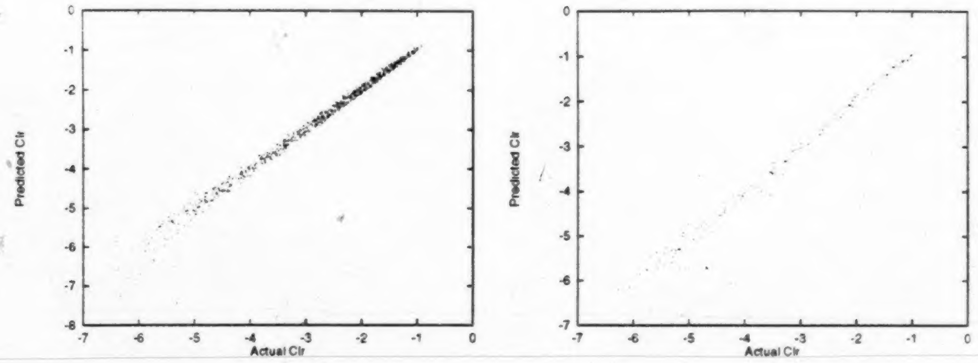


Figure 2.11: C-REG model predictions: training set (left) test set (right)

As in the case of the C-ALN model, the performance of the C-REG model on the test set is not as good as the D-REG model. The reasons for this are mentioned in section 2.4.3. However, the large scatter at low cell loss observed in the case of the C-ALN model is not visible here–the error appears to be fairly constant throughout the entire cell loss rate range in this case.

Figure 2.12 (left) compares D-REG model predictions for $N = 30$—a value not in the training set—with simulation values. Figure 2.12 (right) shows the prediction quality of the C-REG model when $N = 30$. From the slope of the scatter plots, it can be seen that the extrapolation by both models is quite accurate.

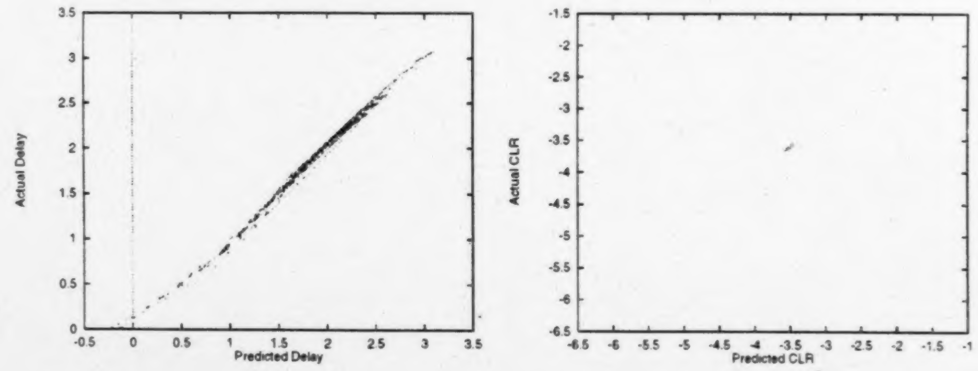


Figure 2.12: Extrapolation for N=30: D-REG model (left) C-REG model (right)

2.5.3 Residual analysis for regression models

Since the non-linear regression models use the least squares principle, the assumptions (section 2.4.4) inherent in the use of least squares apply to these models as well.

In figure 2.13, we can see from the plot of D-REG residuals that the errors are randomly distributed on either side of the mean value. Since the logarithm function is nonlinear for values of the abscissa < 10, we have eliminated these values from the plot. As before, we are more interested in delay predictions at higher delays. There is a small systematic error since the mean residual error is not zero. Also, at low delay, the residuals tend to be negative while at higher delays, the residual error lies above the mean value.

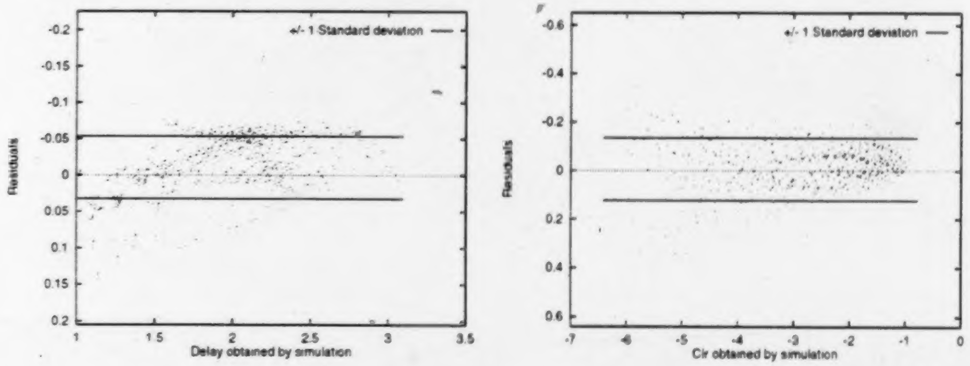


Figure 2.13: Plot of residuals: D-REG model (left) C-REG model (right)

The plot of C-REG residuals shows that the residuals are fairly random up to $Clr = 10^{-6}$. The residuals can be seen to have zero mean since the unit standard deviation limits are equidistant from the horizontal axis. Heteroscedasity (unequal variances) is visible. In the case of the delay model, it manifests itself as large residuals at lower delay values, while in the case of the cell loss model, the residuals can be seen to increase at low cell losses ($< 10^{-4}$).

Model error terms are assumed to follow a normal probability distribution. That this assumption is not unjustified can be observed from histogram plots of C-REG and D-REG model residuals (figure 2.14).

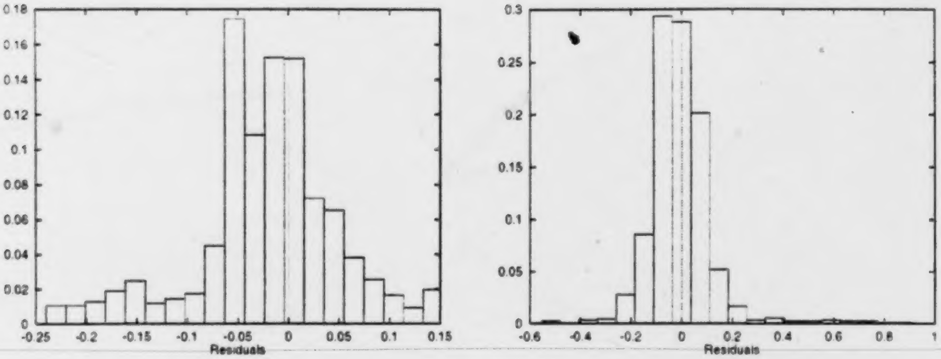


Figure 2.14: Histogram of residuals: D-REG model (left) C-REG model (right)

2.6 The Anick-Mitra-Sondhi model

We review the fluid-flow, infinite buffer model developed by Anick, Mitra and Sondhi [4]. This model, although computationally intensive, was shown in [34] to be significantly more accurate than the approximate model developed by Guerin et al. [15] or the binomial approach in [21].

Let the j-th component of the equilibrium probability distribution vector $\mathbf{F}(x)$ be the probability that j of the N sources are in the *On* state and that the buffer occupancy does not exceed x. Then,

$$\mathbf{F}(x) = \mathbf{F}(\infty) + \sum_{i=0}^{N - \lfloor \frac{BW}{Pcr} \rfloor - 1} e^{z_i x} a_i \phi_i$$

where z_i and ϕ_i are the stable (negative) eigenvalues and eigenvectors associated with the differential equation satisfied by $\mathbf{F}(x)$, and the a_i are constants obtained from boundary conditions. [4] gives an explicit procedure for obtaining the z_i, ϕ_i, and a_i.

2.6.1 AMS cell loss rate model

As a first approximation, the cell loss rate of a finite system with S buffers can be approximated by the probability that the contents x of the infinite buffer exceed S.

This overflow probability is

$$p_S = Pr(x > S) = \sum_{i=0}^{N-\lfloor \frac{RW}{Pcr} \rfloor -1} e^{z_i S} a_i(\mathbf{1}'\phi_i)$$

This is an overestimate, because the mean occupancy of a finite buffer system is lower than that of the infinite buffer. We found empirically that p_S overestimates the real cell loss rate by at least a factor of 2. We therefore use the approximation

$$Clr = p_S/2 \tag{2.7}$$

Figure 2.15 compares the predictions of the C-AMS model on the training and test sets with simulation results.

The reason for the poor performance of the C-AMS model on the test and training sets is two-fold: at high loss rates, the model is inaccurate because the overflow probability used to approximate the cell loss probability is an overestimate. At low loss rates, the model is accurate enough, but the simulation results are not.

Section 2.7 presents a numerical comparison of the AMS results with the ALN and regression models.

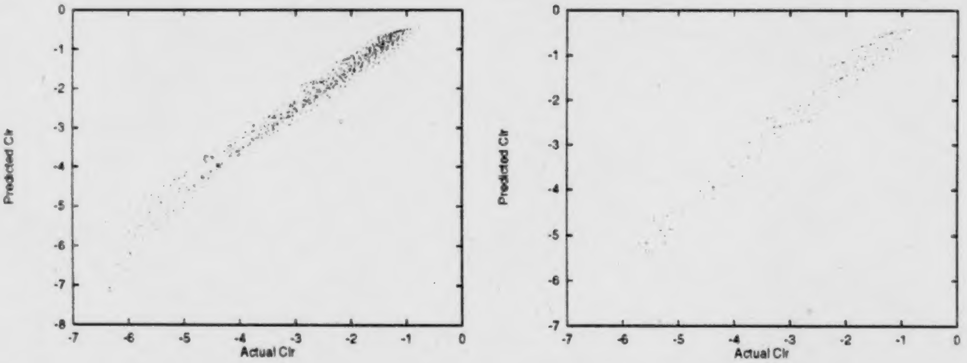


Figure 2.15: C-AMS predictions: training set (left) test set (right)

2.6.2 Extended AMS cell delay model

The mean occupancy of the infinite buffer is given by

$$E[x]_{\infty} = \sum_{i=0}^{N-\lfloor \frac{BW}{Pcr} \rfloor -1} \frac{a_i(\mathbf{1}'\phi_i)}{z_i} \tag{2.8}$$

When there is very high cell loss ($> 10^{-2}$), $E[x]_{\infty}$ grossly overestimates the mean occupancy of a finite buffer and is often much greater than S. We propose approximating delay by the expectation of the buffer occupancy under the condition that this occupancy is no more than S:

$$Del = E[x]_S = \frac{\sum_{i=0}^{S} i \times Pr(x = i)}{\sum_{i=0}^{S} Pr(x = i)} \tag{2.9}$$

Figure 2.16 compares simulation results with the mean delay obtained using equations (2.8) and (2.9) for a few typical cases. While both equations give good results for high bandwidth (i.e., when cell loss is low), (2.9) is significantly more accurate in high cell loss scenarios.

Conditional probability can also be used to approximate jitter (defined as the standard deviation of the cell delay, i.e., the buffer occupancy). Again, the infinite buffer assumption greatly overestimates the variance of a finite buffer, and we approximate $E[x^2]$ by

$$E[x^2]_S = \frac{\sum_{i=0}^{S} i^2 \times Pr(x = i)}{\sum_{i=0}^{S} Pr(x = i)}$$

Then, the jitter is given by

$$jit_S = \sqrt{E[x^2]_S - E[x]_S^2} \tag{2.10}$$

Again, we compare simulation results with the jitter obtained using equation (2.10) and with the infinite buffer jitter (fig. 2.17). As in the case of delay, (2.10) is more accurate when cell loss is high. Note that for our exponential On/Off sources, the jitter has the same order of magnitude as delay.

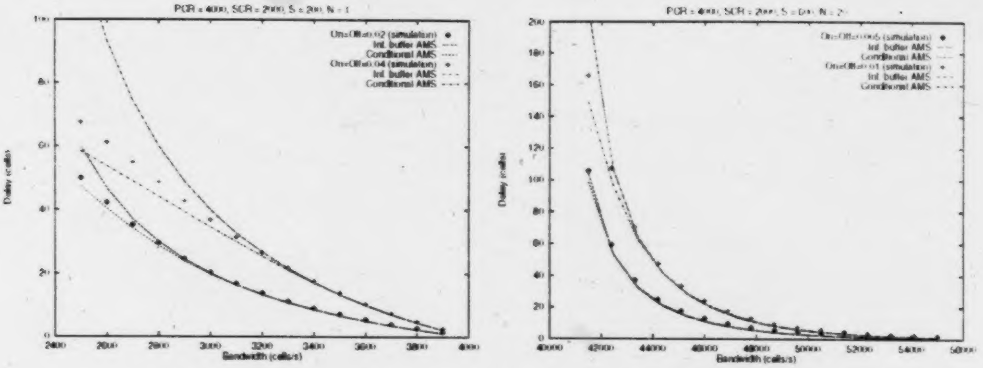


Figure 2.16: Infinite buffer delay and conditional delay

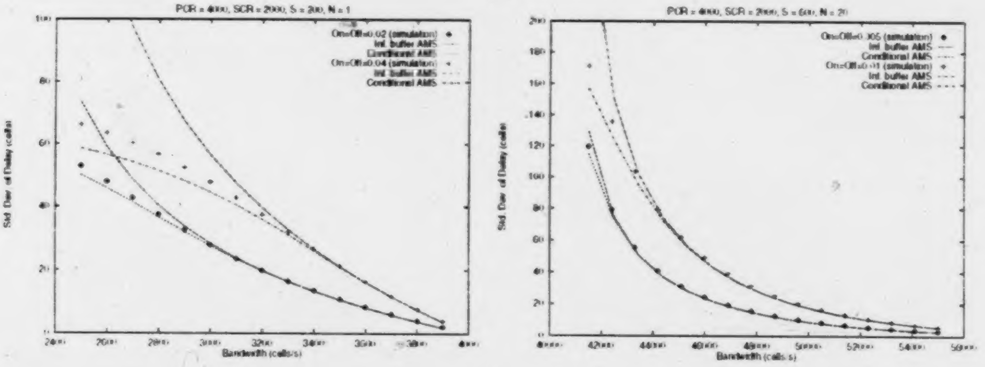


Figure 2.17: Infinite buffer jitter and conditional jitter

Figure 2.18 compares the estimates of the D-AMS model on the training and test sets with values obtained by simulation. It can be seen that the predictions are fairly accurate.

2.7 Comparison of cell delay and loss rate predictions

2.7.1 Cell delay based comparisons

In this section, we compare delay predictions by the D-AMS, D-ALN and D-REG models against the delay values obtained from the simulations.

From the scatter plots presented in sections 2.4, 2.5 and 2.6, it is clear that the

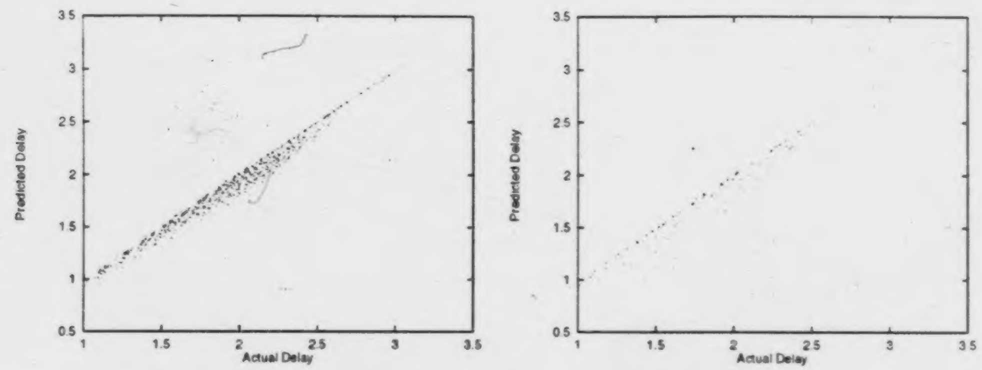


Figure 2.18: D-AMS predictions: training set (left) test set (right)

D-ALN model outperforms the others. The D-REG model ranks a close second. The D-AMS model is rather inaccurate at small to medium delays, probably because the buffer overflow probability is a good approximation to cell loss rate only when the buffer size is large.

Table 2.12 makes a comparison based on the average and maximum *relative* errors[5] in the test and training sets. From the table, we can see that the D-ALN performs the best on test as well as training sets, while the D-REG model comes a close second. The average error of the D-ALN on the test set is almost half that of the D-REG model and only one-third that of D-AMS. The similarity of the average errors on the test and training sets for both the simulation-based models indicates that we have been successful in curtailing model over-training.

Method	*Training set*		*Test set*	
	Maximum error	*Average error*	*Maximum error*	*Average error*
D-AMS	14.6 %	3.3 %	16.1%	3.0 %
D-REG	11.9 %	1.95 %	12.2 %	2.3 %
D-ALN	4.93 %	0.6 %	11.8 %	1.32 %

Table 2.12: Numerical comparison of delay predictions

---

[5] The errors for the delay and cell loss models are computed by comparing against simulation results

2.7.2 Cell loss rate based comparisons

In this section, we compare the cell loss predictions by the C-AMS, C-ALN and C-REG models against the simulation results.

From the scatter plots presented in sections 2.4, 2.5 and 2.6, it appears that the C-ALN model does very well at high cell loss, but its performance declines at lower cell loss. The more consistent C-REG model performs almost as well, and is much more accurate than C-AMS.

Table 2.13 makes a comparison based on the average and maximum *relative* errors in the test and training sets. From the table, we can see that the C-ALN is considerably better than the analytical model, with its average error on the test set being only one-eighth that of the latter. The C-REG model is closer to the C-ALN model in terms of average error, but its maximum error is much larger. This indicates that the simulation-based models can be fairly accurate while offering the benefit of being computationally inexpensive.

Method	*Training set*		*Test set*	
	Maximum error	*Average error*	*Maximum error*	*Average error*
C-AMS	56.5 %	21.7 %	61.7 %	24.0 %
C-REG	17.1 %	3.0 %	32.4 %	5.2 %
C-ALN	19.28 %	1.27 %	19.9 %	3.85 %

Table 2.13: Numerical comparison of cell loss predictions

2.8 Application to effective bandwidth prediction

In the preceding section, we examined three methods of estimating the cell loss and mean delay in our system. We now apply those methods to effective bandwidth computation. We compute the effective bandwidth that satisfies the cell loss requirement, the mean delay requirement, and both. Two requirement sets are considered: the first requirement set (CLR $\leq 10^{-2}$, delay ≤ 100 cell slots) corresponds to a situation with stringent delay and jitter requirements but relatively high tolerance for loss (e.g., real-time video); the second requirement set (CLR $\leq 10^{-4}$, delay ≤ 1000 cell slots) corresponds to the more common case where low loss is required, but delay can be tolerated. Throughout this section, we set $Pcr = 14150$ cells/sec, $t_{ON} = 0.025$ s, and $t_{OFF} = 0.035$ s (this yields an Scr of $0.417 \cdot Pcr = 5896$ cells/sec). The bandwidth is plotted per source and relative to the Pcr, and is thus in $[0.417, 1]$.

Of the three methods, only the ALN could be explicitly inverted; to obtain effective bandwidth from the regression function, and in the AMS model, we used a simple binary iteration approach. We found that generally, accurate results can be obtained in only 5–10 iterations. The declared range of input values for the ALN was $[0.4, 1]$, and the inverted ALN therefore gave values in $[0.4, 1]$, sometimes returning values slightly below the Scr.

2.8.1 Effective bandwidth vs. number of calls

We compute effective bandwidth as a function of the number of calls, which we vary from 1 to 20. The buffer size S is fixed at 5000. The results are plotted in figure 2.19 (requirement set 1 on the left, requirement set 2 on the the right). There are six lines per requirement set: for each of the three approximation methods there is a line for the delay-based effective bandwidth and another for the cell-loss-based effective bandwidth. The effective bandwidth needed to satisfy both delay and cell loss requirements is simply the maximum of the two lines.

We note that the delay requirement is dominant in requirement set 1. The three estimates for delay-based effective bandwidth are very close to each other and show the typical decaying shape (indicating a statistical multiplexing gain). The estimates for the loss-based effective bandwidth are close to the Scr even for $N = 1$ and do not change when N is increased. In this case, it is clear that the ALN underestimates the effective bandwidth, because the estimate is about 5% below the Scr.

For requirement set 2, the loss requirement dominates slightly. As expected, the

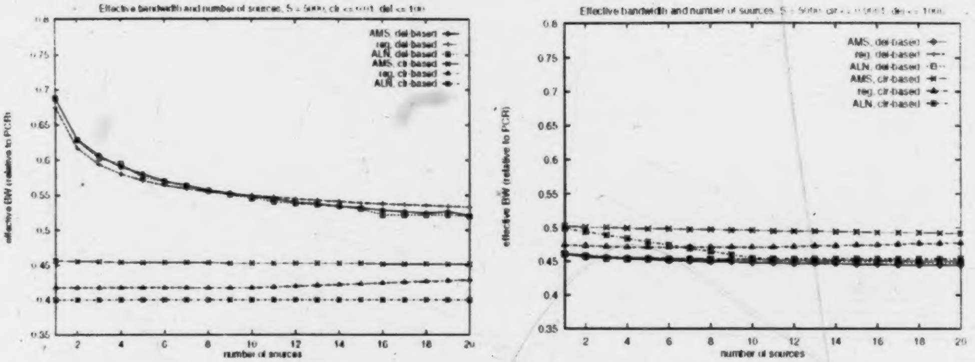


Figure 2.19: Effective bandwidth vs. number of calls

tighter loss requirements increased the loss-based effective bandwidth, and the less stringent delay requirements reduced the delay-based effective bandwidth.

2.8.2 Effective bandwidth vs. buffer size

We compute the effective bandwidth as a function of the buffer size, which we vary from 1 to 10000. The number of calls N is fixed at 10. The results are plotted in figure 2.20 (requirement set 1 on the left, requirement set 2 on the right). Once again, there are six lines per requirement set: for each of the three approximation methods, there is a line for the delay-based effective bandwidth and another for the cell-loss-based effective bandwidth.

The values for delay-based effective bandwidth computed by the three methods exhibit a characteristic *step* form: the delay requirements can be satisfied by any bandwidth if the buffer size is less than the delay requirement; in our figure, the dots are plotted close to the sustained cell rate line. Once the buffer size is relatively large, adding buffers leaves the mean delay, and therefore the delay-based effective bandwidth, at a constant level. Between these two extremes, there is a narrow range of buffer values where an increase in buffer size leads to an increase in mean delay. The reason for this increase is a simultaneous sharp decrease in the cell loss rate, resulting in the delay of cells that would otherwise be dropped.

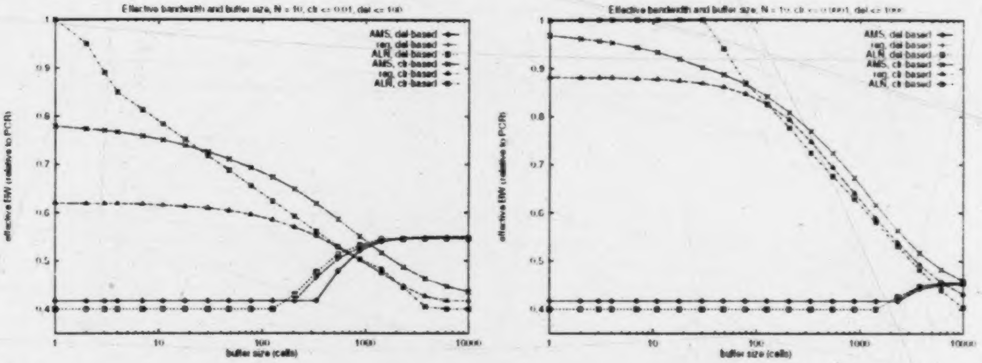


Figure 2.20: Effective bandwidth vs. buffer size

Comparing the graphs for loss-based and delay-based effective bandwidth, we note that the cell loss requirement dominates for small buffer sizes and that the delay requirement dominates for a larger buffer sizes. The crossover point between these two buffer size regions depends on the requirements and on the traffic characteristics. There is no increase in statistical multiplexing gain once the cross-over point has been reached.

The three estimates for loss-based effective bandwidth are divergent when effective bandwidth is high (i.e., when the buffer size is small), and somewhat closer when effective bandwidth is low. This can be explained by the fact that the regression function and the ALN are based on simulations of buffer sizes ≥ 400, and are therefore inaccurate when buffer sizes are small. In the case of delay-based effective bandwidth, the three estimates are very close to each other. The loss-based curves have a shape similar to those obtained in [31].

2.9 Conclusions

Two simulation-based models have been developed using regression analysis and neural network modeling techniques. The models can be used to predict the delay and cell loss rate when a number of bursty sources are multiplexed at a link with finite buffer space. The effective bandwidth requirement to achieve a certain QoS can be easily obtained by inverting the models. We have shown that delay bounds should be considered in addition to loss rate in determining the required service rate.

Unlike other schemes, these models use the number of sources as an input parameter, leading to delay and loss rate computation times that are independent of the number of sources being multiplexed. Though we have used On/Off bursty sources in order to compare the results with other effective bandwidth schemes, the techniques presented here can be extended to complex sources that cannot easily be modeled analytically, e.g., aggregate LAN traffic, MPEG traffic. The models can be developed in a relatively short time compared to analytical models.

The ALN model has an edge over other neural networks in that the same network that is trained to predict cell loss or delay can be inverted to predict effective bandwidth instead. This means that the ALN can make effective bandwidth predictions as quickly as it can make delay or loss predictions. Since the evaluation time is very small—of the order of milliseconds—ALNs are highly suitable for use in real-time CAC. ALN implementation in hardware is easy because the ALNs are composed of AND gates, OR gates and simple linear threshold elements. This can result in an increase in evaluation speeds by an order of magnitude or more.

The regression models, although slower to evaluate than the ALNs, are much faster than the AMS models. They are easier to verify for correctness—under all possible input combinations—as compared to the ALN models. Another advantage of the regression models is that it is possible to obtain an insight into the qualitative behavior of delay and loss rate by inspection of the models—ALN models must be evaluated to retrieve this information. However, regression models become increasingly more difficult to develop as the number of input variables increases.

ALN models can adapt more easily to larger dimensions. The reason is that, in the regression model, the interactions between input variables must be specified explicitly. The ALN modeling technique requires only the relationship between the response variable and each of the inputs to be specified .ALNs learn the relation between the input variables on their own. This simplifies the process of obtaining good models to a great extent.

Chapter 3

Cell scheduling using adaptive feedback control

3.1 Introduction

Consider a server of fixed rate handling cells from two queues. VBR traffic is buffered and statistically multiplexed in one queue while ABR traffic is multiplexed in the other. The service rate for each of the two queues is fixed by CAC. The remaining bandwidth, if any, is utilized by ABR traffic. Within a queue, cells are served in FIFO order.

Several scheduling policies have been proposed to guarantee that each queue receives its allocated share of the bandwidth; for example, packet-by-packet generalized processor sharing (PGPS) [24], virtual clock [39] and weighted fair queuing (WFQ) [10].

3.2 Motivation

The problem with WFQ and other schemes is that they rely on CAC to indicate the correct bandwidth allocation to each queue. If a VBR source does not fully utilize the bandwidth that it requested at call setup time, the extra bandwidth allocated to the VBR queue results in the QoS being better than needed. It would be preferable for ABR traffic to utilize the excess bandwidth. Also, bit rate fluctuations in the aggregate VBR traffic can result in QoS fluctuation above the desired level—resulting in QoS violation even when the sources are conforming to their traffic descriptors.

Thus, the observed QoS for the VBR queue can vary above and below the aggregate requirement. Since the sources are conforming, the long-term QoS is likely to converge to the guarantee; however, since call durations are arbitrary, such a guarantee is of questionable value.

The goal of our scheme is to modulate the bandwidth allocated to a queue in such a way that the QoS delivered to the aggregate traffic remains at its desired level, in spite of individual source fluctuations and/or parameter variations. CAC specifies the *mean level* about which the service rate is to fluctuate, as well as the *bounds* for the service rate modulation.

One of our design objectives was generality; we demonstrate in section 3.9 that our scheme works without modification for bursty traffic generated by three very different source models, using very general information about the traffic source in each case. Since the scheduler tries to maintain the QoS at all times, there is no need to restrict the QoS guarantee to calls of long duration.

The models in the literature for generating aggregate traffic are either very specific (for example, Ethernet traffic) or complex (multi-state MMPP); it is very difficult to model the aggregate traffic at a switch deep within a wide area network in a simple, yet general, way. In such a situation, rather than assuming that the aggregate traffic follows a known distribution, we assume that nothing is known about it. Therefore, the only information that the scheduler uses is the average rate of the aggregate traffic (the sum of the average rates specified in the individual traffic descriptors) and the time-scale information (also derived from the traffic descriptors).

Our scheme, like WFQ, assumes the availability of a CAC algorithm that allocates the requisite amount of bandwidth to a group of statistically multiplexed sources. Section 2.1 describes a few bandwidth allocation schemes that can be used for this purpose. The neural network and regression models presented in chapter 1 are also suitable for this purpose.

The bandwidth specified by the CAC in the above schemes is usually based on the assumption that the traffic is generated by bursty On/Off sources with exponentially distributed on and off periods. This may not be an accurate model—for example, [27] suggests that real sources may have a fixed on period. Hence, it is quite likely that the bandwidth allocated by CAC will be inaccurate and this in turn will affect the performance of WFQ. Our scheme on the other hand will modulate the service rate to achieve the QoS and hence can deliver the required performance even when the CAC-specified bandwidth is incorrect.

Rather than a hard partitioning of the link bandwidth among the different queues, we assume that the partitions are movable to a small extent, specified as a percentage deviation (positive and negative) from the bandwidth allocated by CAC. The actual queue service rate is modulated within these limits to maintain the QoS at the desired value. A similar feedback control approach is proposed by Pitsillides et al. [29]. However, they chose to regulate the flow of the traffic entering the queue. The problem with this scheme is that its performance is adversely affected by the distance between the source and the queue, making it unsuitable for wide area networks. Moreover, since the traffic entering the queue may be generated by multiple sources, flow control information must be sent through the network to each of these sources. Another disadvantage is that the selection of a sampling time for the controller (see section 3.8.1) is dictated by the source that is farthest away, leading to unnecessarily slow control of nearby sources. We avoid these problems by *modulating the service rate at the queue itself.* This allows us to make the controller sampling time as small as necessary to ensure quick response to disturbances.

3.3 Background

In this section, we present a general overview of adaptive feedback control and online system identification.

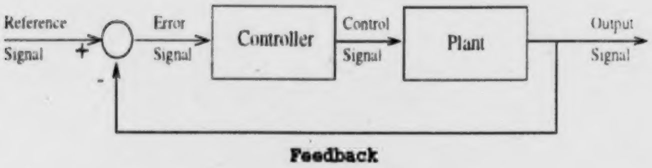


Figure 3.1: Feedback control system

3.3.1 Feedback control

A computer-controlled system[1] is shown in figure 3.1. The output $y(t)$ from the system to be controlled (henceforth known as the *plant*), is a continuous-time signal. The desired value for $y(t)$ is called the reference signal $y^*(t)$[2]. The difference between the desired value of the plant output $y^*(t)$ and the actual plant output $y(t)$ is the error signal $\epsilon(t)$. Disturbances act on the system and prevent it from responding deterministically to a given input.

The error signal is converted into digital form by an analog-to-digital (A/D) converter by sampling $\epsilon(t)$ at a specified rate, corresponding to the time instants t_k, k=0,1,.... The computer processes the sampled signal $\epsilon(t_k)$ using an algorithm (the *control* algorithm) to generate a new discrete-time signal $u(t_k)$. A digital-to-analog (D/A) converter, operating at the same sampling frequency as the A/D converter, converts $u(t_k)$ into its analog representation $u(t)$. Also, $u(t_k)$ is converted into a continuous time signal whose values at the sampling instants correspond to the values of $u(t_k)$, but whose value between samples is constant. This signal is then fed to the input of the plant. Thus the ***output*** of the controller is the plant *input*.

The choice of the control algorithm is crucial to the performance of the controller. Basically, the control algorithm uses the error signal to determine the plant input that will cause the plant output to move as close as possible to the desired value, in spite of the system disturbances. In order to do this, the controller must have an idea of the degree to which a given input signal affects the output. This cause and effect relationship between the plant input and output can be modeled in two ways: an explicit transfer function relationship between the input and output, or an implicit state-space transfer model in which the input and output are related by intermediate state variables of the plant. The latter approach is preferred to deal with multi-input, multi-output (MIMO) systems; since we will be considering only single-input, single-output (SISO) systems, we will restrict our attention to input-output transfer function models. A methodology for obtaining the system model is presented in section 3.3.2 and strategies for designing the controller algorithm are described in section 3.3.3.

---

[1] In control terminology, the system to be controlled is often referred to as the *plant*. We will use these terms interchangeably

[2] For example, in a room whose temperature is controlled by a thermostat, the reference signal would be the desired temperature.

3.3.2 System identification

In this section, we will describe how a model of the plant can be built using only sampled plant input and output data, an approach known as *black box* modeling. This is useful when very little *a priori* information about the plant is available. Plant models can be obtained under three different circumstances. In the first situation (figure 3.2), the plant is perturbed by a carefully chosen input signal. The corresponding plant outputs are recorded at the sampling instants. The collection of input and output data can be then analyzed off-line to obtain an open-loop model of the plant.

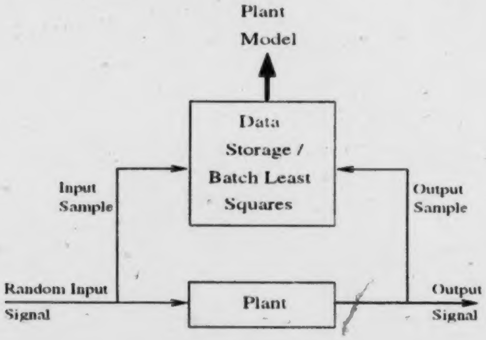


Figure 3.2: Open-loop system identification

Alternatively, plant input and data can be obtained while the system is under feedback control. The plant input must be chosen so that it excites all the modes of the system, i.e. it should contain information over the entire bandwidth range of the system under test. During feedback control, the plant input signal is generated by the controller and therefore may not adequately excite the plant. This will result in a poor or incorrect model identification. Hence an independent dither signal must be introduced in the feedback loop to provide sufficient (or *persistent*) excitation. In this case, as in the case of open loop identification, the collected data is analyzed at the end of the experiment, resulting in the *batch identification* procedure described below.

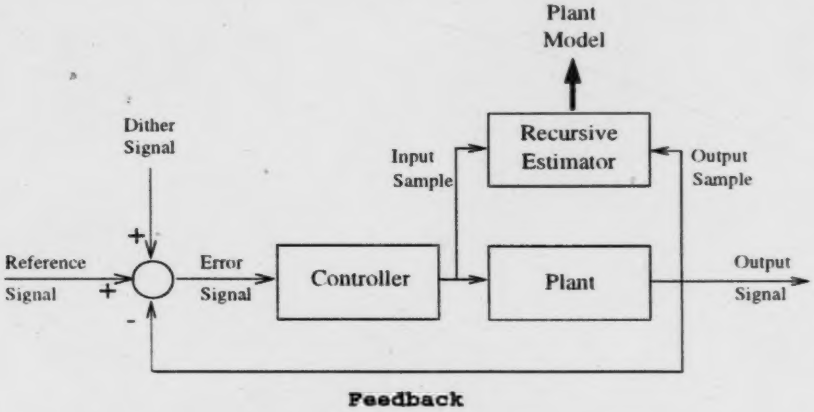


Figure 3.3: Closed-loop system identification

The last situation concerns *online* model identification (figure 3.3 right). When the plant is started up, it is left to run for some time in open-loop mode, during which time it is excited by a random signal, the range of which is chosen carefully to ensure that the plant output is acceptable. The plant outputs are recorded along with the inputs at the sampling instants. When sufficient data has been recorded, a model of a pre-specified order (and therefore a given number of unknown coefficients) is identified. From then on, at every sampling instant, the new sample of the plant input and output is used to update the model calculated at the previous sampling instant. This continues after the plant is placed under closed-loop control. Such a procedure of recursive identification is of great importance in adaptive control (section 3.3.4).

3.3.2.1 Modeling the input-output relationship

Basically, a system contains a set of elements which are interrelated to fulfill a particular function ([17]). On receiving a stimulus, the response of the plant is governed by its internal structure. The input signal is transformed (or filtered) into the system response by the internal elements (figure 3.4 left). If an input applied at time t_k appears at the output of the system at time t_{k+d}, then d is called the system delay, which is the time lag as a result of the filtering process.

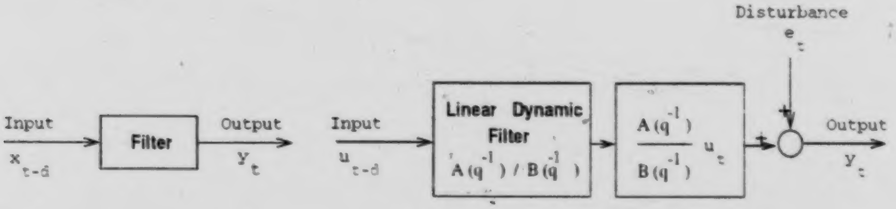


Figure 3.4: System representation : Filter (left) Transfer function (right)

Thus, the objective of investigating the input-output relationship is to model the system filter. Because of disturbances that affect the system, the system output is related in a non-deterministic manner to its input. Therefore, the input-output relationship must be modeled in a stochastic sense (figure 3.4 right).

The system input and output are treated as stochastic processes and are assumed to be discrete-time. The transfer function approach allows us to model the linear system with few parameters. In the SISO case, the model is of the form

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})} u(t-d) + e(t) \tag{3.1}$$

In the above equation, q^{-1} is the discrete-time, backward shift operator defined by

- $q^{-1} y(t) = y(t-1)$
- $q^{-2} y(t) = y(t-2)$

and so on. $A(q^{-1})$ and $B(q^{-1})$ are polynomials in q^{-1} of orders n and m, respectively. ϵ is a random disturbance signal with zero mean and unit variance. It is usually the case that $n > m$; The larger of m and n is called the model order. The system delay is d time samples.

Equation 3.1 indicates that the system output at time t is a function of m past inputs, n past outputs and a random noise component, $\epsilon(t)$. This structure represents an autoregressive, exogenous input (ARX) model of the input-output relation.

3.3.2.2 Batch model identification

We assume that N input-output values of plant data are available for modeling purposes. We would like to find the ARX model of order a that satisfactorily explains the input-output relationship. We assume that the system delay, d, is one sampling period.

We can rewrite equation 3.1 in the following way[3]:

$$y(t) = \mathbf{x}^T(t)\,\theta + \epsilon(t) \tag{3.2}$$

where θ is the vector of unknown coefficients of the A and B polynomials

$$\theta^T = [-a_1, \ldots, -a_n, b_0, \ldots, b_m]$$

and $\mathbf{x}(t)$ is a regression vector consisting of past measured input and output values

$$\mathbf{x}^T(t) = [y(t-1), \ldots, y(t-n), u(t-1), \ldots, u(t-m-1)]$$

Now let us assume that equation 3.2 is an exact representation of the system, and that we wish to determine the vector θ of true system parameters from available data. In order to do this, we assume a model of the system of the correct structure:

$$y(t) = \mathbf{x}^T(t)\,\hat{\theta} + \hat{\epsilon}(t) \tag{3.3}$$

---

[3] $\mathbf{x}^T$ denotes the transpose of matrix $\mathbf{x}$.

where $\hat{\theta}$ is a vector of adjustable model parameters and $\hat{\epsilon}(t)$ is the corresponding modeling error at time t. Our aim is to select $\hat{\theta}$ so that the modeling error over all the data points is minimized in some sense.

Since we have N input-output values,

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} \mathbf{x}^T(1) \\ \mathbf{x}^T(2) \\ \vdots \\ \mathbf{x}^T(N) \end{bmatrix} \theta + \begin{bmatrix} \hat{\epsilon}(1) \\ \hat{\epsilon}(2) \\ \vdots \\ \hat{\epsilon}(N) \end{bmatrix}$$

To be able to estimate the $m+n$ unknowns uniquely, the number of equations N must be greater than or equal to the number of unknowns. In the noise-free case, the equation can be solved as a set of linear equations in $m+n$ unknowns. The resulting modeling errors will be identically zero. In the presence of noise, we must have $N >> m+n$ and use an alternative procedure to reduce estimation errors caused by the noise. The technique most widely used in this context is *linear least squares*.

Let us rewrite equation 3.3 as

$$\mathbf{y} = \mathbf{X}\,\hat{\theta} + \hat{\mathbf{e}} \tag{3.4}$$

where

$$\mathbf{y}^T = [y(1), \ldots, y(N)]$$

and

$$\hat{\mathbf{e}}^T = [\hat{\epsilon}(1), \ldots, \hat{\epsilon}(N)]$$

and

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T(1) \\ \mathbf{x}^T(2) \\ \vdots \\ \mathbf{x}^T(N) \end{bmatrix}$$

which in turn can be written as

$$\hat{\mathbf{e}} = \mathbf{y} - \mathbf{X}\,\hat{\theta}$$

The principle of least squares was first used by the mathematician Gauss to investigate the motion of the heavenly bodies. According to him [7], the unknown parameters of a model should be chosen in such a way that:

> *The sum of the squares of the differences between the actually observed and computed values multiplied by numbers that measure the degree of precision is a minimum*

In other words, let us choose an estimate $\hat{\theta}$ that minimizes J, the sum of squares of errors:

$$J = \sum_{t=1}^{N} \hat{\epsilon}^2(t) = \hat{\mathbf{e}}^T \, \hat{\mathbf{e}}$$

That is,

$$\begin{aligned} J &= (\mathbf{y} - \mathbf{X}\,\hat{\theta})^T\,(\mathbf{y} - \mathbf{X}\,\hat{\theta}) \\ &= \mathbf{y}^T\,\mathbf{y} - \hat{\theta}^T\,\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\,\mathbf{X}\,\hat{\theta} + \hat{\theta}^T\,\mathbf{X}^T\,\mathbf{X}\,\hat{\theta} \end{aligned}$$

To find the minimum value of J, we set the derivative of J with respect to $\hat{\theta}$:

$$\frac{\partial J}{\partial \hat{\theta}} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\theta = 0 \tag{3.5}$$

If the second derivative matrix

$$\frac{\partial J^2}{\partial \hat{\theta}^2} = 2(\mathbf{X}^T\mathbf{X})$$

is positive definite[4], the least squares estimator for the parameter vector (equation 3.5) is

$$\hat{\theta} = [\mathbf{X}^T\mathbf{X}]^{-1}[\mathbf{X}^T\mathbf{y}] \tag{3.6}$$

Therefore, given the data matrix $\mathbf{X}$ consisting of past input and output samples, and the vector $\mathbf{y}$ of current output samples, we can find the unknown coefficients of the ARX model relating the plant input-output. Note that the estimate of θ in equation 3.5

---

[4] A symmetric matrix is positive definite if its eigenvalues are positive

was obtained by choice of a specific criterion—the least squares criterion. Use of a different criterion will result in a different set of parameter estimates. As mentioned before, the least squares estimator is one of the most popular criteria.

The matrix $(\mathbf{X}^T\mathbf{X})^{-1}$ is of great importance since it gives a direct measure of the variability of the parameter estimates. We can now see the importance of the persistent excitation mentioned in section 3.3.2. If the plant excitation $u(t)$ is not adequate, then the matrix $\mathbf{X}^T\mathbf{X}$ will be singular. Hence, the covariance of the parameter estimates will be infinite, indicating that the model is very poorly identified. Thus, in adaptive control, it is important that the change in $u(t)$ is sufficient to avoid a rank-deficient $\mathbf{X}^T\mathbf{X}$ matrix. If the system is noise corrupted, then fed back noise may provide sufficient excitation, otherwise excitation may be provided either by varying the reference signal or by injecting an independent dither signal into the feedback loop.

3.3.2.3 Online model identification

The batch model identification technique presented in the previous section must be modified if it is desired to identify an upto-date model, online, at each sampling instant. In this case, the observations are submitted to the identification process sequentially, instead of all at once as in the batch case. At any sampling instant, it is possible to apply the batch identification algorithm to the input-output data collected up to that time.

At the next instant, the input-output data is augmented by the new sample of plant input and output and the batch algorithm is rerun to estimate the current model. This can easily be seen to be very inefficient. Instead, the computations are arranged so that the parameter estimates after N observations can be used to get the estimates after $N+1$ observations. This procedure is referred to as *recursive* identification.

In this scheme [37], new input-output data becomes available at each sampling instant. At any time t, the model obtained at the previous sampling instant, $\hat{\theta}(t-1)$, summarizes the past input and output of the plant. This model is used to obtain an estimate $\hat{y}(t)$ of the current output which is compared with the actual output $y(t)$ to generate an error $\epsilon(t)$. The error causes a model update, correcting $\hat{\theta}(t-1)$ to the new value $\hat{\theta}(t)$. This recursive "predictive-corrector" form allows significant savings in computation, since it eliminates the need to save all the past input and output data.

The algorithm ([37]) for recursive least squares identification (RLS) is as follows : At time step $t+1$:

1. Form $\mathbf{x}(t+1)$ using the new data sample

2. Form $\epsilon(t+1)$ using

$$\epsilon(t+1) = y(t+1) - \mathbf{x}^T(t+1)\hat{\theta}(t)$$

3. Form $\mathbf{P}$(t+1) using

$$\mathbf{P}(t+1) = \mathbf{P}(t)\left[\mathbf{I}_m - \frac{\mathbf{x}(t+1)\mathbf{x}^T(t+1)\mathbf{P}(t)}{1+\mathbf{x}^T(t+1)\mathbf{P}(t)\mathbf{x}(t+1)}\right]$$

where $\mathbf{P}$(t+1) is the covariance matrix at time $t+1$, i.e

$$\mathbf{P}(t+1) = [\mathbf{X}^T(t+1)\mathbf{X}(t+1)]^{-1}$$

where $\mathbf{I}_m$ is the $m \times m$ identity matrix.

4. Update $\hat{\theta}(t)$

$$\hat{\theta}(t+1) = \hat{\theta}(t) + \mathbf{P}(t)\mathbf{x}(t+1)\epsilon(t+1)$$

5. At the end of the sample period, go to step 1.

3.3.2.3.1 Initializing the recursive estimator

The RLS algorithm requires an initial estimate $\hat{\theta}(0)$ of the parameter vector. Since the choice of the initial estimate is not crucial to convergence of the algorithm, we can choose $\hat{\theta}(0) = \mathbf{0}$.

Another object that requires initialization is the covariance matrix $\mathbf{P}(t)$. The values of the elements of this matrix reflect our uncertainty with respect to the unknown parameters. If we have no idea of the true values, then a large initial covariance is suitable, while if the initial parameter estimates are close to their actual values, a small covariance matrix should be used. Usually, $\mathbf{P}(0)$ is chosen as:

$$\mathbf{P}(0) = r\mathbf{I}_m$$

where r is in the region of 100 to 1000 for a large covariance matrix, and in the range 1–10 for a small initial setting.

Finally, before the estimator can be started up, the data vector $\mathbf{x}(t)$ must be full. For a model with A and B polynomials of degree n and m, respectively, the sampling process must run for τ steps (assuming a system delay of one sampling period):

$$\tau = max(n, m + 1)$$

before the estimator can be started.

3.3.2.3.2 Adaptive RLS The RLS algorithm will converge to the true parameters of the model if the excitation is adequate and the model structure has been chosen correctly. However, if the true parameters are time-varying, then the estimator needs to be able to track the parameters. One way of getting the recursive estimator to adapt is the ***forgetting factor*** technique.

The forgetting factor $\lambda \in (0, 1]$ is used to progressively reduce the emphasis placed on past information. The least squares criterion minimizes the cost function

$$J = \sum_{i=1}^{N} \hat{\epsilon}^2(i)$$

in which all values of $\hat{\epsilon}^2(i)$, $i = 1, \ldots, N$ are equally important. The forgetting factor approach uses a differential weighting scheme in which older information is given lower emphasis in the following cost function

$$J' = \sum_{i=1}^{N} \lambda^{N-i} \hat{\epsilon}^2(i)$$

The popularity of the forgetting approach stems from the ease with which it can be incorporated into the RLS algorithm. The only change necessary to the RLS algorithm is in step 3 which now becomes

$$\mathbf{P}(t+1) = \lambda^{-1}\mathbf{P}(t)\left[\mathbf{I}_m - \frac{\mathbf{x}(t+1)\mathbf{x}^T(t+1)\mathbf{P}(t)}{\lambda + \mathbf{x}^T(t+1)\mathbf{P}(t)\mathbf{x}(t+1)}\right]$$

λ is usually chosen to lie in the interval [0.95, 1].

3.3.2.3.3 Rigorous recursive least squares (RRLS) estimation The RLS algorithm presented in the previous section suffers from the following disadvantages:

- The results depend on the initial choice of the covariance matrix
- The results are not the same as those obtained by the batch least squares method.

The RRLS algorithm [19] overcomes the above problems. The process is modeled by an ARX equation of the form

$$y_k = A_1\, y_{k-1} + \cdots + A_n\, y_{k-n} + B_1\, u_{k-1} + \cdots + B_m\, u_{k-m} + c + \epsilon_k \qquad (3.7)$$

For a single-input, single-output (SISO) system, y_k is the process output sampled at time k, u_k is the process input at time k and $\{\epsilon_k\}$ is white or colored noise with zero mean. Estimation of a constant offset c means that the sampled y and u values need not be mean-centered and auto-scaled as must otherwise be done. The order of the model is determined by the larger of m and n.

Define

$$\mathbf{b}_k = \mathbf{P}_{k-1}\mathbf{x}_k, c_k = \lambda + \mathbf{x}_k^T\mathbf{b}_k, \mathbf{a}_k = \mathbf{Q}_{k-1}\mathbf{x}_k$$

and let the initial conditions be

$$\hat{\theta}_0 = 0, \mathbf{P}_0 = r\mathbf{I}, \mathbf{Q}_0 = \mathbf{I}$$

The algorithm is defined in the following way (for a SISO system):

$$\left\{\begin{array}{l} \mathbf{g}_k = \mathbf{a}_k(\mathbf{a}_k^T\mathbf{a}_k)^{-1} \\ \hat{\theta}_k = \hat{\theta}_{k-1} + \mathbf{g}_k(y_k - \mathbf{x}_k^T\hat{\theta}_{k-1}) \\ \mathbf{P}_k = (\mathbf{P}_{k-1} - \mathbf{g}_k\mathbf{b}_k^T - \mathbf{b}_k\mathbf{g}_k^T + c_k\mathbf{g}_k\mathbf{g}_k^T)/\lambda \\ \mathbf{Q}_k = (\mathbf{Q}_{k-1} - \mathbf{g}_k\mathbf{a}_k^T \end{array}\right. \qquad \text{if } \mathbf{a}_k \neq 0$$

$$\left\{\begin{array}{l} \mathbf{g}_k = \mathbf{b}_k c_k^{-1} \\ \hat{\theta}_k = \hat{\theta}_{k-1} + \mathbf{g}_k(y_k - \mathbf{x}_k^T\hat{\theta}_{k-1}) \\ \mathbf{P}_k = (\mathbf{P}_{k-1} - \mathbf{g}_k\mathbf{b}_k^T)/\lambda \\ \mathbf{Q}_k = (\mathbf{Q}_{k-1} \end{array}\right. \qquad \text{if } \mathbf{a}_k = 0 \text{ (ordinary RLS)}$$

3.3.3 Controller design

The aim of a feedback controller is two-fold: to modify the dynamic response of the plant and to reduce the sensitivity of the plant output to disturbances. An additional objective may be to reduce the output fluctuations as a a result of parameter variations.

A number of feedback control algorithms can be linked with the batch or recursive estimators discussed in the previous section. They belong to three basic types:

- Pole assignment control
- Minimum variance (MV) control
- Long range predictive control (LRPC)

The aim of pole assignment control is to exactly match the closed loop characteristic equation of a feedback system to a specified form. The specified form is based on classical control design requirements stated in terms of desired frequency and/or transient response. However, in achieving the desired response, it is possible that the noise rejection ability is impaired resulting in large output variances. Hence the controller design must also include a variance reducing strategy.

In the minimum variance controller, the control design rule is based upon optimization techniques. At each time step, the output of the controller is chosen such that it minimizes the variance of the plant output. LRPC controllers retain the optimization framework of (MV) control but avoids the problems associated with MV, such as sensitivity to incorrect system delay specification and inability to control non-minimum phase[5] systems.

We present the minimum-bias LRPC scheme developed by [19]. Assume that a model of the plant is available, either obtained online by a recursive estimator or statically obtained from a batch identification experiment.

From equation 3.7, we can see that the plant output at time k, y_k, is a function of

$$y_k = F(y_{k-1}, \ldots, y_{k-n}, u_{k-1}, \ldots, u_{k-m}, c)$$

Define sets S_k^{fr} and S_k^{fo} as

$$S_k^{fr} = \{y_k, y_{k-1}, \ldots, y_{k+1-n}, u_{k-1}, \ldots, u_{k+1-m}, c\}$$

and

$$S_k^{fo} = \{u_k, u_{k+1}, \ldots, u_{k+i-1} : i \geq 1\}$$

At time k, the predicted model output $\hat{y}_{k+i}$ at time $k+i$ is a linear combination of the free response and the forced response. The former is the prediction based on the elements of S_k^{fr} (past inputs and outputs) assuming that the elements of S_k^{fo} (inputs for $i \geq k$) are zero. The latter is the prediction based on the elements of S_k^{fo}, assuming that the elements of S_k^{fr} are zero. That is,

$$\hat{y}_{k+i} = \hat{y}_{k+i}^{fr} + \hat{y}_{k+i}^{fo}$$

where $\hat{y}_{k+i}^{fr}$ is the predicted free response and $\hat{y}_{k+i}^{fo}$ is the predicted forced response.

For some integer $N \geq 1$, the control horizon, define

$$\hat{y}_{k,N}^{fr} = \begin{bmatrix} \hat{y}_{k+d}^{fr} \\ \hat{y}_{k+d+1}^{fr} \\ \vdots \\ \hat{y}_{k+d+N-1}^{fr} \end{bmatrix}, \hat{y}_{k,N} = \begin{bmatrix} \hat{y}_{k+d} \\ \hat{y}_{k+d+1} \\ \vdots \\ \hat{y}_{k+d+N-1} \end{bmatrix}$$

---

[5] The roots of the B polynomial are outside the unit circle

and

$$u_{k,N} = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-1} \end{bmatrix}$$

where d is the system delay.

Therefore, the predicted plant output, N steps into the future, is:

$$\hat{y}_{k,N} = \hat{y}^{fr}_{k,N} + C_{k,N} u_{k,N}$$

where $C_{k,N}$ is a matrix determined by the parameter vector ($\hat{\theta}_k$ for an recursively estimated model, $\hat{\theta}$ for a static model).

The problem of finding the minimum-bias control can be expressed as the following multi-objective optimization problem:

$$\begin{cases} \text{minimize } J_{MB} = \|\hat{y}^{fr}_{k,N} + C_{k,N} u_{k,N} - y^{*}_{k,t}\| \\ \text{subject to } u_{k,t} \in \mathcal{U}_{k,t} \end{cases}$$

where $\mathcal{U}_{k,t}$ is the set of admissible control values, specified as an upper and lower bound on $u_{k,t}$. The vector $y^{*}_{k,t}$ consists of values of the reference signal at the N future sampling instants.

3.3.4 Adaptive control

The controller design described in the previous section can be used along with a static model obtained by off-line batch least squares identification to form a conventional feedback control system. Such a design works well under the following assumptions about the plant to be controlled:

- The plant is time-invariant, i.e., the parameters of the plant (the coefficients of the A and B polynomials) do not vary with time. In practice, however, system parameters do often change over a period of time.

- The system is linear. That is, if an input $x_1(t)$ produces an output $y_1(t)$ and an input $x_2(t)$ produces an output $y_2(t)$, then the response to an input $a\,\cdot$

$x_1(t) + b \cdot x_2(t)$ will be $a \cdot y_1(t) + b \cdot y_2(t)$, for any choice of constants a, b. In practice, conventional controller design still works well even when applied to slightly nonlinear processes.

When the above requirements are not met, a static model of the plant will not suffice; the model will be out of date because of parameter variation or because of a shift in the operating point. In this case, adaptive feedback control is necessary.

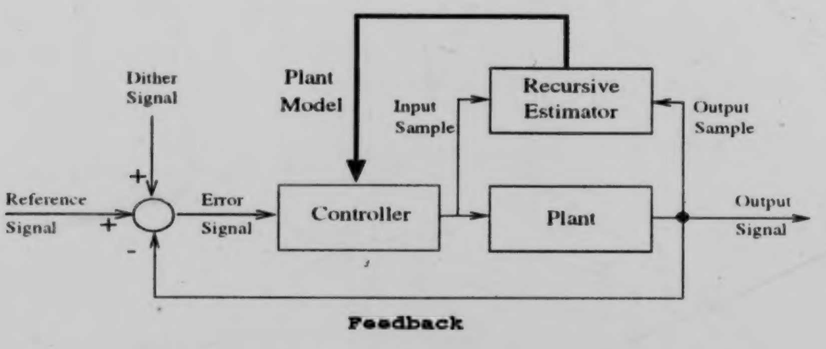


Figure 3.5: Adaptive feedback controller

Figure 3.5 shows an adaptive feedback control system. This differs from the conventional system in figure 3.3.1 in the following way: an online model of the system is estimated recursively at each sampling instant. This model, rather than the static model used in the conventional system, is used by the feedback controller to make the control decision.

While the terms adaptive and self-tuning control are sometimes treated as synonymous, we would like to make a distinction between them as follows [7]: *self-tuning* is motivated by a desire to achieve control of a process with constant or very slowly changing parameters. It is usually taken to apply to the initial automatic tuning of a controller. *Adaptive* control, on the other hand, is viewed as a mechanism for continuous adjustment in the face of varying parameters.

3.4 The queuing system

The system to be controlled consists of the queue for the VBR traffic class (**rt-vbr** according to the ATM Forum [8]) at one output port of an ATM switch and a server for that queue. The capacity of the server is equal to the bandwidth allocated to the VBR queue by CAC. This allows us to ignore the ABR traffic in studying the feedback control scheduler.

The CAC algorithm also specifies upper and lower constraints on the service rate. The lower bound on the service rate is governed by the rule that the service rate of the queue cannot fall below the mean arrival rate, if the queue is to remain stable. The upper bound can be arbitrarily specified as a percentage (> 100 %) of the mean service rate or determined by any suitable rule. For the duration of a simulation experiment, the number of sources generating aggregate traffic into the queue is assumed to remain constant.

3.4.1 Source traffic descriptor

At call admission time, each source passes a traffic descriptor to the CAC module at the switch. This descriptor describes the traffic that will be generated by the source of the connection duration. The traffic descriptor is expected to consist of ([8])

Peak Cell Rate (PCR) This traffic parameter specifies an upper bound on the rate at which the traffic can be submitted on a connection.

Sustainable Cell Rate (SCR) The SCR is an upper bound on the average rate of the traffic on an ATM connection, over time scales which are long relative to those for which the PCR is defined. To simplify matters, we assume that the source specifies the average rate for the duration of the connection.

Burst Tolerance (BT) This is the maximum number of cells that can arrive back-to-back, while still conforming to SCR.

We assume that the CAC passes the following information to the virtual scheduler:

- The peak rate of the aggregate traffic—the sum of the peak rates of the individual sources
- The average rate of the aggregate traffic—the sum of the average rates of the individual sources

This ensures that the scheduler is not dependent on the more detailed information about the expected aggregate traffic; rather the scheduler makes minimal assumptions about the traffic entering the queue.

3.4.2 Quality of service

While the traffic generated by the sources need not be identical, the fact that the sources belong to the same traffic class implies that they have the same quality of service requirements.

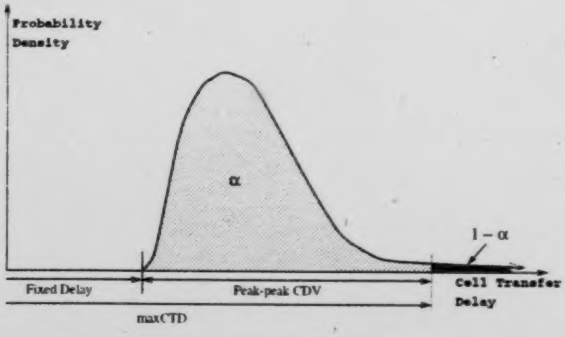


Figure 3.6: ATM Forum CTD probability density model

The ATM Forum has specified the following QoS parameters for real-time VBR traffic ([8]):

Cell Loss Ratio (CLR) This parameter is defined over the lifetime of the connection as

$$CLR = \frac{Cells\ lost}{Total\ cells\ transmitted}$$

Maximum Cell Transfer Delay (maxCTD) Figure 3.6 illustrates the probability density function of the end-to-end cell transfer delay (CTD). The fixed delay includes propagation delays, transmission delays and fixed components of switch processing delay. Cell Delay Variation (CDV) is induced by switch buffering and cell scheduling. The maximum CTD for a connection is specified in terms of the α quantile of the CTD. This means that only a fraction $1 - \alpha$ of the cells will have a delay greater than maxCTD. The ATM Forum recommends that CLR be used as an upper bound on α.

Peak-to-peak CDV (ppCDV) This parameter indicates the amount of allowable *jitter* and is specified as the difference between maxCTD and the fixed delay.

From figure 3.6, we can see that the sum of ppCDV and fixed delay gives the α quantile of CTD. For example, if the source makes the following specifications:

- At least 99 % of the cells should have a delay less than maxCTD, i.e., $\alpha = 0.99$.
- The desired peak-to-peak CDV is 1 ms.

Then, if the fixed end-to-end delay is 0.1 ms, the QoS requirement is that at least 99 % of the cells should have an end-to-end transfer delay of less than $1 + 0.1 = 1.1$ ms.

Since the system that we will be studying consists of a single queue, we will assume the fixed delay to be zero. We will also use $\alpha = 1-\mathrm{CLR}$. In that case, the QoS requirement (assuming the same QoS specification for all the sources, as mentioned earlier) can be stated as:

> The fraction of the cells given by $1-\mathrm{CLR}$ should have a delay less than ppCDV[6].

From now on, the QoS will be specified as a tuple

$$< \mathrm{CLR}_{qos}, ppCDV_{qos} >$$

---

[6] If ppCDV is specified in time units, it can be converted to cell units using the mean service rate.

Also, we will see section 3.9.4 that the average cell delay is a function of the service rate allocation made by CAC. For this reason, *the feedback control system is only responsible for maintaining the cell loss rate and delay jitter at the desired levels.*

3.5 System model

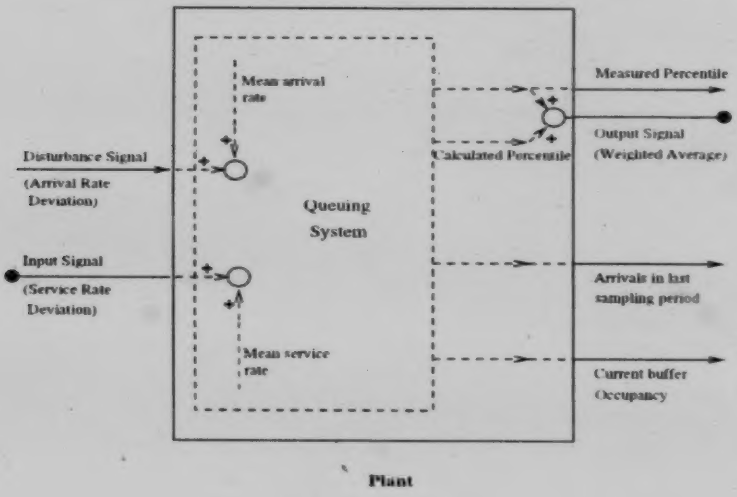


Figure 3.7: Queuing system

The previous section presented the queuing system and the quality of service requirement to be achieved for the aggregate traffic. Now we will see how this system is transformed into another SISO system that is controllable. We will also see how the QoS specification can be transformed into a reference signal used to control the system. Figure 3.7 describes how and where the queue fits into the feedback control system shown in figure 3.5.

3.5.1 Choosing the plant output signal

The desired QoS consists of a cell loss rate and delay jitter specification (section 3.4.2). Ensuring that the QoS requirements are met entails monitoring the actual loss rate and jitter and comparing them with the reference values. To simplify the control system, we would like to have a plant with a single output and therefore a single reference signal. This means that the output of the plant should describe the actual loss rate as well as the jitter. One such quantity is the α quantile of the buffer occupancy.

If we can control the queuing system in such a way that the α quantile of the buffer occupancy is maintained at a desired level K_{ref}, then by choosing $\alpha = \mathrm{CLR}_{desired}$ and $K_{ref} = pp\mathrm{CDV}_{qos}$, we can ensure that the QoS requirements are met.

3.5.1.1 Measuring the buffer quantile

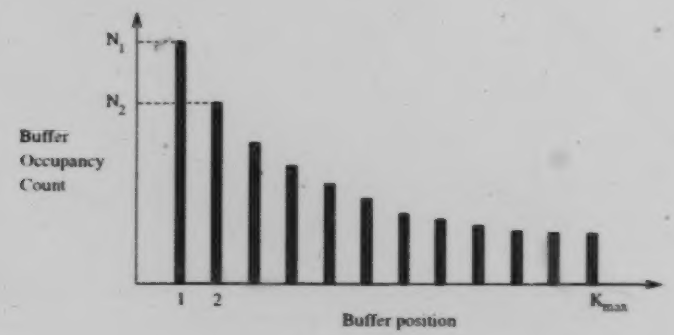


Figure 3.8: Buffer quantile histogram

Figure 3.8 shows a snapshot of the buffer occupancy histogram during a hypothetical run. The number of cells, N_i that occupy position $i, 1 \leq i \leq K_{max}$ in the queue are recorded. K_{max} is the index of the cell buffer such that $N_{K_{max}+1} = 0$; i.e, K_{max} is the highest position occupied at any time during the run.

Define the frequency of buffer position i as

$$f_i = \frac{N_i}{\sum_{i=1}^{K_{max}} N_i}$$

and the cumulative frequency as

$$c_i = \frac{\sum_{j=1}^{i} N_j}{\sum_{i=1}^{K_{max}} N_i}$$

The cumulative frequencies satisfy the following property:

$$c_i > c_{i-1}, \forall i = 1, \dots, K_{max} \text{ and } c_{K_{max}} = 1$$

Let j be the buffer position such that

$$c_j \geq \alpha \text{ and } c_{j-1} \leq \alpha$$

then the α quantile, K_{alpha}, is obtained by interpolation in the following way[7]:

$$K_{alpha} = j - \frac{c_j - \alpha}{c_j - c_{j-1}}$$

3.5.1.2 Calculating the buffer quantile

A method for approximating the probability distributions of stationary statistics—in particular, the mean and α quantile—in FIFO single server queues is described in [1]. The method is applicable to semi-Markov queues: in such queues the tail of the waiting time distribution is negative exponential. That is, the tail can be approximated by ce^{s^*t}, where s^* is negative. Arrival processes with independent and identically distributed increments (e.g. Poisson) are special cases of a semi-Markov process and hence the method can be used to study such traffic as well.

An important application of the method is when the *net input process* to the queue forms a stationary, ergodic, Gaussian discrete-time process. The net input process is defined as the difference between the arrival and service processes. Since the service rate in ATM networks is deterministic, the service process has zero variance and is specified by its mean alone.

The Gaussian process can be used to model any traffic encountered in practice since its mean, variance and auto-covariance function can be matched to any process. Moreover, the aggregate traffic resulting from the multiplexing of a number of independent streams is likely to be a Gaussian process. Since our focus is on the aggregate traffic at an interior ATM node, the assumption of Gaussian traffic is quite reasonable.

The predicted α-quantile is a function of the mean, variance, and auto-covariance sum of the net arrival process defined as

---

[7] Although the buffer positions in the queue are indexed by integers, $K_{alpha} \in \Re^1$

$$Y_n = A_n - B_n$$

where A_n is the amount of work entering the queue during the n^{th} sampling interval (of duration T_{sample}), and B_n is the amount of work processed by the server during the same interval. For a deterministic server, $B_n = T_{sample}/\tau_{cell\ time}$, where $\tau_{cell\ time}$ is the time to service a cell.

The following quantities are defined in [1]:

$$s^* = \frac{2m}{\sigma^2 + 2S} \quad \text{and} \quad \tilde{c} = \frac{erfc\left(\frac{-m}{\sigma\sqrt{2}}\right) - e^{(u-m)s^*}\, erfc\left(\frac{2u-m}{\sigma\sqrt{2}}\right)}{s^*\left(\frac{\sigma\sqrt{2}}{\sqrt{\pi}}e^{-\left(\frac{u^2}{2\sigma^2}\right)} - u\, erfc\frac{u}{\sigma\sqrt{2}}\right)}$$

where $u = \sigma^2 s^*/2$. m is the mean value of Y_n, i.e., $m = E\{Y_n\} = E\{A_n\} - E\{B_n\}$. σ^2 is the variance of Y_n, i.e., $\sigma^2 = Var\{Y_n\} = Var\{A_n\}$ since B_n is deterministic. S is the auto-covariance sum of Y_n defined as the sum of the covariances of Y_n for all lags > 1:

$$S \triangleq \sum_{k=1}^{\infty} Cov(Y_n, Y_{n+k}) \tag{3.8}$$

These quantities are used to compute the α-quantile of the buffer occupancy as

$$K_{alpha} = \frac{1}{s^*} ln\left(\frac{1-\alpha}{\tilde{c}}\right) \tag{3.9}$$

As mentioned in [2], computing the auto-covariance sum (S) on line is not practical. For this reason, we compute an approximate value for the p^{th} quantile using only the mean and variance of the net arrival process (substituting $S = 0$ in the above equation for s^*).

In such a case, using this approximation as the feedback value for control may result in the actual quantile being quite different from the computed value. Therefore, the feedback signal that we actually use, $y^*(t)$ is computed as

$$y^*(t) = \begin{cases} y(t) & \text{if } |\, y(t) - y_{act}(t)\, | < 0.2 * y_{act}(t) \\ 0.9 * y_{act}(t) + 0.1 * y(t) & \text{otherwise} \end{cases}$$

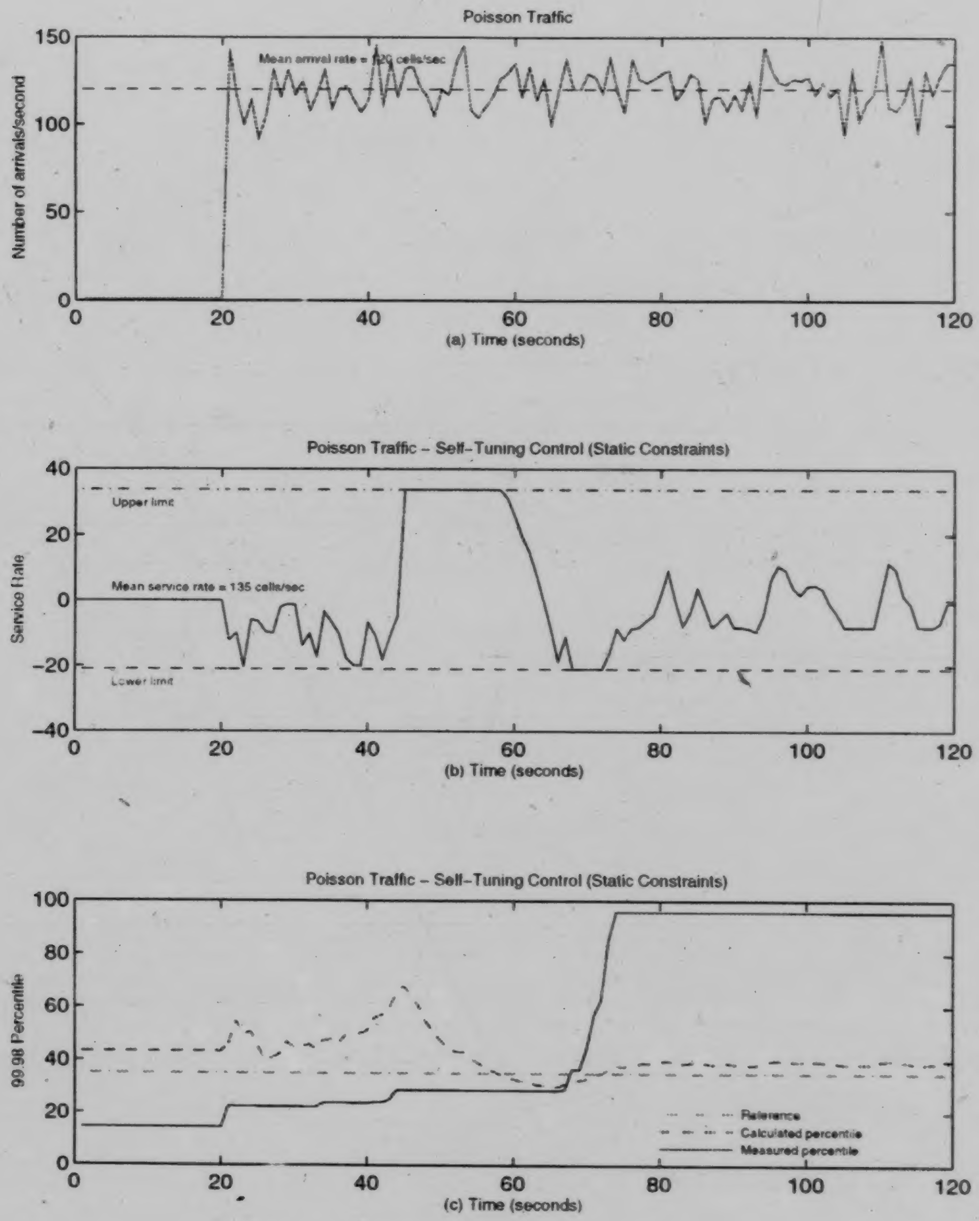


Figure 3.9: Feedback control of calculated quantile

where the weighted average of the actual value $y_{act}(t)$ and computed value $y(t)$ is used whenever the relative error is greater than 20%.

Figure 3.9 shows the performance of the controller when the output signal is the calculated quantile alone. The traffic entering the queue is Poisson. The QoS requirement is specified as

$$< 0.0002,\ 30\ cells >$$

which can be translated to mean that the 99.98% of the buffer occupancy is to be held at 30 cells. Hence the reference signal is chosen as 30.

It can be seen that while the calculated quantile is maintained quite close to the reference level (figure (c)), the actual (measured) quantile is not. This is the reason for using the weighted average instead. From a control perspective, the calculated quantile is expected to give the controller a sense of direction, i.e, an idea of whether to increase or decrease the input to take the plant output closer to the reference level. The measured quantile is expected to give the control an idea of the *gain*—the magnitude of change in the input necessary to take the output to the desired value—as well as an idea of the dominant *time constant*—the speed of response to an input change.

In [29], the calculated quantile is used as the plant output signal. The control system in that case guarantees that the long term QoS will equal the desired value. However, during the transient period, the estimated quantile (and the actual quantile) may greatly exceed the reference level.

The use of the weighted average has the advantage that the QoS requirements are not violated at any time, removing the restriction of large call duration. Also, because of the use of a smaller controller sampling time (section 3.8.1), the QoS is taken to the desired level in a much shorter time.

3.5.2 Calculation of the reference signal

Since the plant output signal is the α-quantile of the buffer occupancy, the reference signal must be the desired value of the α-quantile, which also corresponds to the desired QoS.

The translation process can be illustrated as follows. Let the QoS requirement be specified as

$$< 0.0002, 30\ cells >$$

This can be translated to mean that the $1 - 0.0002 = 0.9998$ quantile (or equivalently, the 99.98 percentile) of the buffer occupancy is to be held at 30 cells[8]. Hence the reference signal is chosen as 30.

3.5.3 Choosing the plant input signal

The buffer distribution can be altered by varying either the arrival process or the service process, since the net input process is a function of both. One of the guidelines in choosing the input signal is that the the output must respond quickly to a change in the input.

The sources generating traffic into the queue may be located at large distances away. If we choose to regulate the sources in an attempt to control the arrival process (flow control), the response delay is increased by one round-trip-time. Fast sampling enables the controller to monitor the plant output closely and take quick action to nullify the effect of disturbances. As mentioned in section 3.2, flow control prevents us from choosing small control sample times—more importantly, it prevents us from choosing a sampling time based on the time scale of the aggregate traffic alone; the propagation delays to the various sources must be known as well. Flow control also results in protocol complexity due to the need for sending resource management cells to the sources.

---

[8] The time for transmitting a single cell is based on the service rate allocated by CAC

For these reasons, we choose to modulate the service rate. The mean service rate as well as upper and lower limits on the service rate are specified by CAC. For example, the CAC algorithm may allocate a mean bandwidth of 40 Mb/sec with upper and lower limits of 20 and 50 Mb/s, respectively. The output of the controller (plant input) would be a signal whose magnitude lies in $[-20, 10]$ Mb/s. The value of this signal at a sampling instant would be internally added by the plant to the mean service rate parameter to obtain the actual service rate for the next sample duration (figure 3.5).

3.6 Traffic models

In this section we discuss the various models that are used to generate aggregate traffic into the queue.

3.6.1 Poisson traffic

The inter-arrival time between cells generated by the Poisson model is exponentially distributed with a specified mean $\bar{t} = 1/\mathrm{SCR}$, where SCR is the mean arrival rate. While the assumption of Poisson traffic is an oversimplification for a single source, it is not so in the case of aggregate traffic for the following reasons.

The On/Off model is frequently used to model voice and video traffic sources. The source model has an *On* and an *Off* state. When in the *On* state, the source generates cells at a deterministic rate of PCR cells/s; no cells are generated during the *Off* period. The *On* and *Off* periods are exponentially distributed with means t_{ON} and t_{OFF}, respectively

When the number of sources is large, the aggregate traffic due to bursty On/Off type sources will generally be smooth, because of overlapping bursts from the different sources. On the other hand, the aggregate traffic due to a number of Poisson sources will still be Poisson and therefore, bursty. Since we are trying to control the QoS for the aggregate traffic, the assumption of Poisson arrivals is not an oversimplification.

but it can be viewed as a worst-case assumption, because Poisson traffic does not aggregate at all.

Also, as shown in section 3.7, the queuing system supplied with Poisson traffic is very non-linear and hence complex from the point of feedback control.

3.6.2 Batch-Poisson traffic

We would like to test the performance of the controller on aggregate traffic with greater burstiness than the Poisson process. As mentioned in [38], the batch Poisson process is the limiting case for the aggregate traffic due to a number of multiplexed On/Off bursty sources. It represents the most bursty pattern that a number of On/Off sources can generate, for a given average arrival rate and burst length.

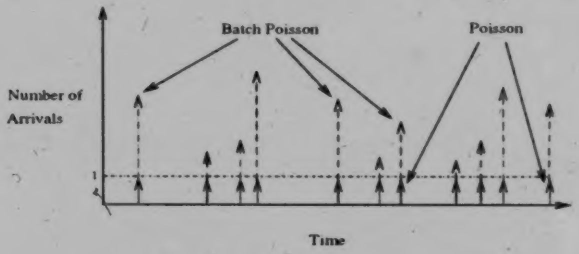


Figure 3.10: Batch Poisson traffic model

In this model, bursts consisting of multiple cells arrive at the queue instantaneously following a Poisson process with an average rate λ (figure 3.10). The burst size L is assumed to be geometrically distributed (or, if the burst size is much greater than one cell, an exponential distribution can be used). The average of rate cell traffic entering the queue is $SCR = \lambda * L$.

3.6.3 Self-similar traffic

While the batch Poisson model generates adequately bursty traffic, it suffers from one deficiency (along with the Poisson model)—the generated traffic is not *long-range dependent*.

Results in [12, 18, 26] have shown that

- It is possible to distinguish between actual network traffic and traffic generated by the widely employed Markovian models discussed previously.
- In contrast to traditional packet traffic models, aggregate traffic streams are statistically *self-similar* or *fractal*; i.e. the bursty nature of real traffic is retained even when aggregated over several orders of time scale. Long-range dependence (LRD) is a manifestation of the self-similarity of the aggregate traffic.

A recent study [12] has shown that the presence of LRD in traffic completely alters the tail of queue waiting times. One of the conclusions drawn is that the use of models—Poisson and batch Poisson models among others—with *short-range dependence(SRD)* for buffer dimensioning may lead to overly optimistic network performance predictions and inadequate network resource allocations.

In what follows, we present some terms and definitions used in characterizing self-similar traffic.

Definition

Consider a stochastic process $X = \{X_t, t = 1, 2, \cdots\}$ where

- all $\{X_t\}$ are drawn from the same distribution with mean μ and finite variance σ^2, and
- $\gamma_k = Cov(X_t, X_{t+k})$ is independent of t for all integers k. The autocorrelation function $r(k)$ is defined as $r(k) = \gamma_k/\sigma^2$.

Then the process $\{X_t\}$ is said to be *covariance stationary*. We assume that X has an autocorrelation function of the form[9]

$$r(k) \sim k^{-\beta} L(t), 0 < \beta < 1 \qquad (3.10)$$

---

[9] here and throughout the text, ~ indicates that the expressions on the two sides are asymptotically proportional to each other

where we assume that $L(t)$ is asymptotically constant.

For each $m = 1, 2, 3, \cdots$, let $X^{(m)} = \{X_k^{(m)}, k = 1, 2, \cdots\}$ denote the new covariance stationary series obtained by averaging the original series X over non-overlapping blocks of size m:

$$X_k^{(m)} = \frac{X_{km-m+1} + \cdots + X_{km}}{m}, k \geq 1$$

and $r^{(m)}(k)$ be the autocorrelation function of the new series $X^{(m)}$.

The original process is called exactly *second-order self-similar* with parameter H, if for all $m = 1, 2, \cdots$

$$Var(X^{(m)}) = \sigma^2 m^{-\beta}, \quad r^{(m)}(k) = r(k), k \geq 0$$

where $Var(X)$ is the variance of the time-series X. That is, X is second-order self-similar if the autocorrelation function of any series aggregated from X is the same as that of the original series X. H is commonly known as the *Hurst* parameter and is related to β above by $H = 1 - \beta/2$. For short-range dependent process, $H = 0.5$ while for self-similar processes, $0.5 < H < 1$.

Self-similarity manifests itself in a number of equivalent ways:

1. The variance of the sample mean decreases more slowly than the reciprocal of the sample size, i.e., $Var(X^{(m)}) \sim cm^{-\beta}$, where c is a finite positive constant. For traffic processes with only SRD, $Var(X^{(m)}) \sim cm^{-1}$. This also means that the log($Var(X^{(m)})$)–log(m) plot will have a slope = -1 for processes with SRD and a slope between 0 and -1 for self-similar processes.

2. The autocorrelations decay hyperbolically rather than exponentially fast resulting in an non-summable autocorrelation function $\sum_k r(k) = \infty$ when r(k) satisfies relation 3.10. Intuitively, this means that there is a high degree of correlation even at large lags. However, for SRD processes, $\sum_k r(k) < \infty$ since for these processes, $r(k) \sim a^{|k|}, 0 < a < 1$.

Because of (2) above, the auto-covariance sum calculated using equation 3.8 may be large. Hence the assumption made in section 3.5.1.2 that $S = 0$ will result in a calculated buffer quantile that is greatly in error. Even worse, the calculated quantile may not respond to service rate variations in a way that will give the controller an idea whether the input must be increased or decreased to take the plant output to the reference value. Thus, the controller performance is expected to deteriorate when the aggregate traffic is highly self-similar ($H > 0.7$).

For queues with Markovian traffic as input, the queue length distribution has a dominant tail which can be approximated by a negative exponential. This asymptotic form is important in traffic engineering; it forms the basis of equivalent bandwidth schemes ([15, 11]). Moreover, the buffer quantile in section 3.5.1.2 was calculated under the assumption that the distribution of the unfinished work could be approximated by the tail distribution. However, [12] shows that the tail distribution in self-similar queues is not negative exponential; consequently a new approach is required to calculate the equivalent bandwidth [22] and the buffer quantile [3].

3.6.3.1 Self-similar traffic generator

To test the performance of the controller in the face of aggregate self-similar traffic, we use the traffic generator described in [25]. The algorithm generates a class of self-similar traffic known as *fractional Gaussian noise (FGN)*.

FGN: $X = \{X_k : k \geq 0\}$ with parameter $H \in (0.5, 1)$ is a stationary Gaussian process with mean μ, variance σ^2, and autocorrelation function

$$r(k) = \frac{| k+1 |^{2H} - | k |^{2H} | k-1 |^{2H}}{2}, k > 0$$

and is exactly second-order self-similar.

The method works as follows : given the power spectrum of the FGN process, a sequence of complex numbers z_i corresponding to this power spectrum is calculated; z_i can be thought of as a frequency-domain sample path. Using the Discrete

Time Fourier Transform (DTFT) method, the time-domain counterpart of z_i, x_i, are obtained. By construction, x_i has the same power spectrum as z_i. Because autocorrelation function and power spectrum form a transform pair, x_i is guaranteed to have the auto-correlation properties—which, as mentioned above, distinguish self-similar traffic—of an FGN process. The method is accurate and fast.

The model inputs consist of the desired mean, variance and Hurst parameter. The output is a sequence of values (the sample path) representing the number of arrivals in a chosen time period (bin); this must be the same interval for which the desired mean and variance are specified.

However, the queue simulation takes inter-arrival times as inputs, not arrival counts. As a first step, therefore, the real-valued arrival counts are converted to integers; each represents the number of arrivals occurring during that bin. The simplest way to obtain inter-arrival times is to divide the arrivals uniformly over the bin, although this tends to underestimate the burstiness.

3.7 Open-loop identification

In this section, we study the characteristics of the queuing system by performing the following experiments (using Poisson traffic):

- To obtain the steady-state characteristics of the system as a function of the mean service rate, the mean arrival rate was fixed at 120 cells/sec. The mean service rate was varied from 125 cells/sec to 165 cells/sec corresponding to a variation in utilization from 0.73 to 0.96. The simulation time of each experiment was sufficient to ensure that the calculated quantile reached a stable value. The final value of the calculated quantile was recorded in each case. Figure 3.11 plots the calculated and measured quantiles as a function of the mean service rate. It can be seen that at high buffer utilization, the calculated and measured quantiles are non-linear functions of the mean service rate.

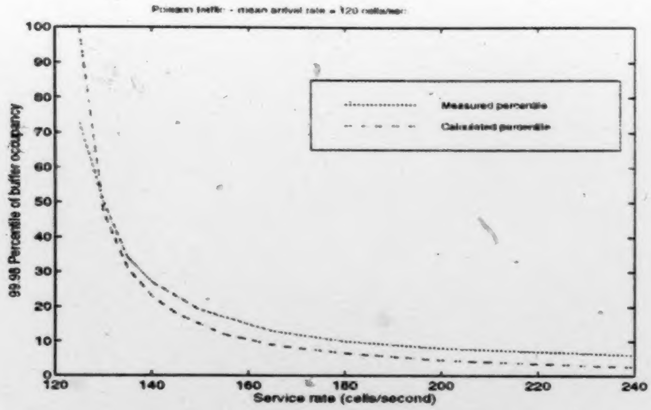


Figure 3.11: Open loop steady-state response

- In the second experiment, we record the response of the queuing system to step inputs of varying sizes. The mean arrival rate was 120 cells/sec, as before while the mean service rate was fixed at 150 cells/sec. In each case, after the simulation was run for 200 seconds, the service rate was changed to a new value, obtained by adding a positive (or negative) step to the mean service rate. The variation in the calculated quantile as a result of the step input was recorded. Figure 3.12 plots the calculated quantile as a function of time, for different step sizes. The response for each step size is normalized w.r.t the step size before plotting. The following observations can be made from the figure:

1. The steady state values of the buffer occupancy quantile are higher when a negative step is applied. This implies that the process gain is greater for decreases in service rate (a small decrease in service rate can cause a larger change in quantile than an increase of the same magnitude). This non-linearity is noticeable in figure 3.11 as well.

2. For both positive and negative step changes, the normalized steady state quantile levels depend on the magnitude of the step, showing that the gain is a function of the step size.

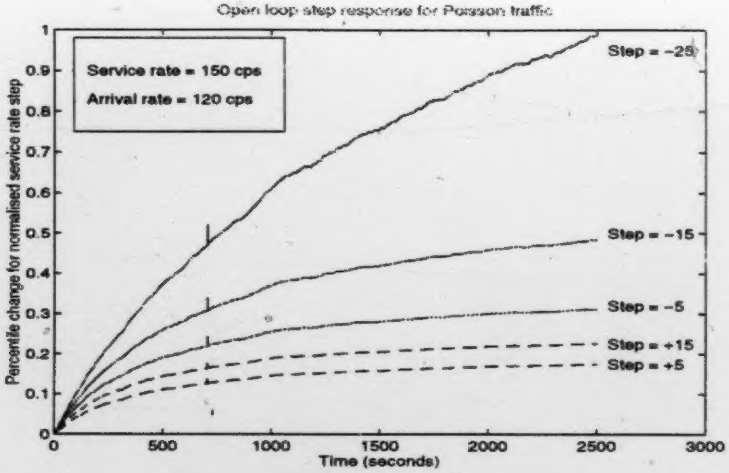


Figure 3.12: Open loop step response

3. We notice that the time to reach the steady state is lower for negative step sizes, as can be seen by the higher slope. Thus the response to a negative step has a lower time constant.

4. The time constant for negative (or positive) step sizes is dependent on the magnitude of the step change.

From 1 and 2, we can conclude that the process has a gain-type non-linearity. (A linear constant-gain process should have the same values for normalized steady state levels, independent of the step size). From 3 and 4, we infer that the process also has non-linear response times. Thus, we can see that the queuing system with Poisson input is non-linear at high utilization and therefore, not trivial from a control point of view.

- To obtain an idea of the model dimensions required to obtain a good prediction of the system response, we carry out an open loop identification test (section 3.3.2.1). The system is excited by a square wave that is internally added to the mean service rate parameter to obtain the instantaneous service rate. Figure 3.13 shows the calculated quantile obtained in response to the excitation. The following observations can be made from the figure:

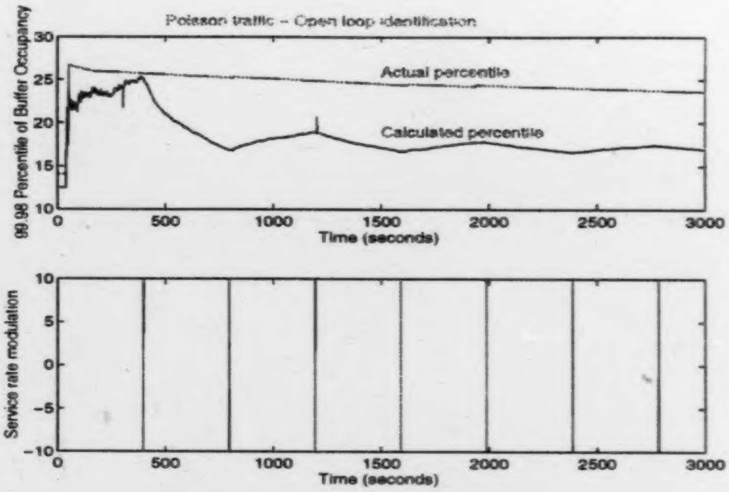


Figure 3.13: Open loop identification

1. The calculated quantile responds within one sample time (0.1 second) to a change in the service rate. This suggests that the system delay d in equation 3.1 can be chosen as 1 (if the sampling time is 0.1 seconds).

2. The magnitude of the response decreases progressively as a function of time, even though the input excitation remains the same. This is the result of the gain and time constant non-linearities mentioned before. As a consequence, it is not possible to describe the behavior of the system with a single input-output model with fixed parameters. Such a model, if obtained using the batch least squares procedure, will result in values for the parameters that are averaged over the entire simulation time, implying that the model will not be accurate enough at any given instant. Besides, the model parameters obtained will depend on the length of the simulation run. It is partly for this reason that we do not attempt feedback control with a fixed model—rather we use adaptive control where the model iden-

tified is used for only a short period of time before being replaced by an updated model.

3.7.1 Choosing the plant sampling time

As mentioned in section 3.5.1.2, in order to estimate the buffer quantile, time is divided into sampling intervals of duration T_{sample}. The difference between the amount of work (cells) arriving during the i^{th} sample time and the number of cells leaving the queue is a random variable X_i. The net arrival process A is defined as the stochastic process $A = \{X_i : 0 \leq i < \infty\}$.

The mean μ and variance σ^2 of the net arrival process are used to calculate the buffer quantile. Theoretically, the accuracy of the estimated quantile is independent of the sampling interval. However, in practice, the mean and variance are calculated from a finite number of samples, The calculated mean $\hat{\mu}$ and variance $\widehat{\sigma^2}$ are therefore estimates of the actual values. Larger sampling times lead to larger values of X_i and correspondingly larger deviations from the true mean and variance. This effect is even more predominant when the arriving traffic is self-similar because in this case, the variance of the amount of work arriving in an interval n has a larger than linear growth ($n^{2H}, 0.5 < H < 1$) for a considerable range of n values.

For the above reasons, we will choose the sampling interval so that $\mid \mu \mid$ is of the order of a few cells. For a mean arrival rate of 120 cells/sec and a service rate of 150 cells/sec, we chose the sampling time T_{sample} to be 0.1 seconds ($\mid \mu \mid= 3$). This is the default value for T_{sample} in all the experiments that we performed.

3.8 Closed-loop control

Figure 3.14 shows the feedback control system implemented using SIMULINK®, a simulation program[10] for dynamic systems, a computing environment for numeric

---

[10] SIMULINK is a registered trademark of The Mathworks, Inc. (URL : http://www.mathworks.com)

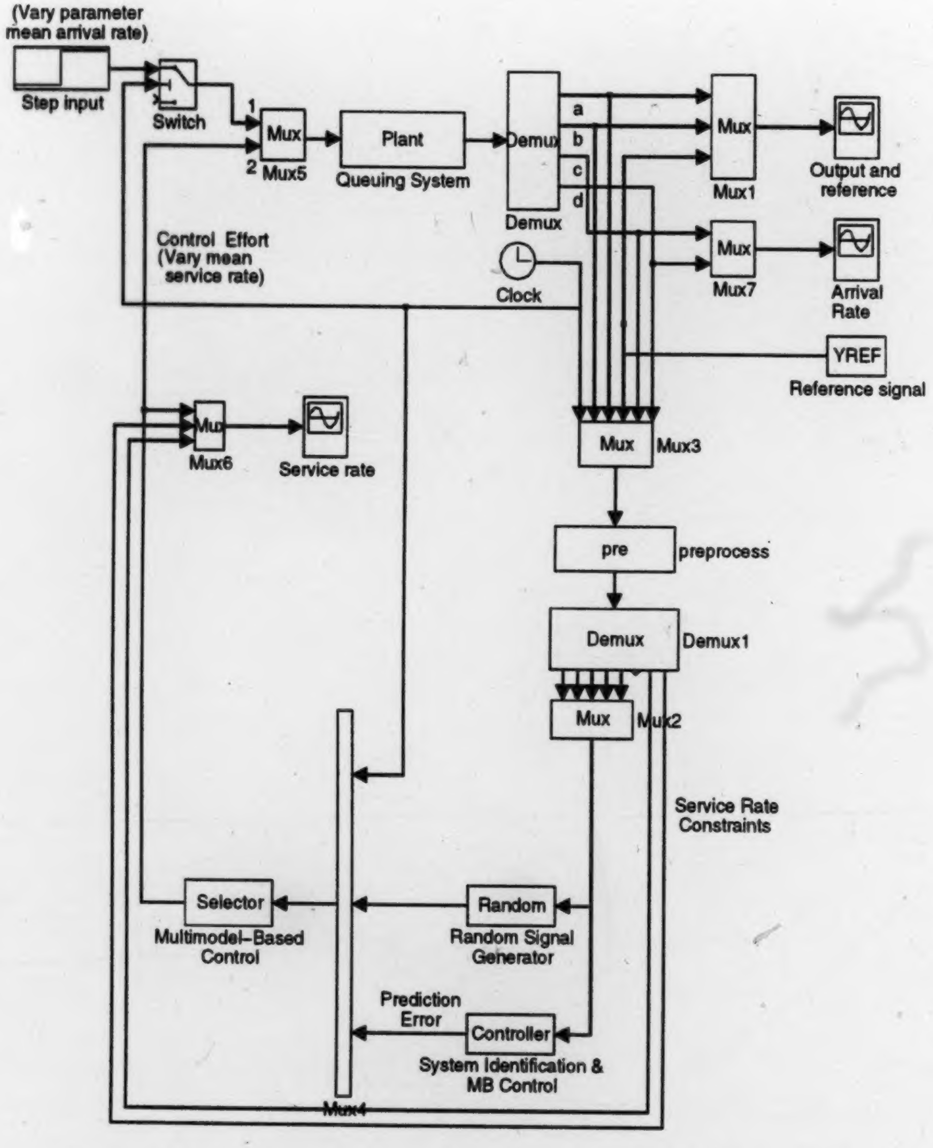


Figure 3.14: SIMULINK feedback control system (modified from [19])

computation and visualization. The queuing system was implemented in C++ to speed up the simulation. The other elements of the system were either built-in blocks available within SIMULINK® or written from scratch using **MATLAB** script files. The functionality of some of the blocks is discussed below:

- The queuing system block `Plant` simulates a queue with four different traffic types: Poisson, batch Poisson, On/Off and self-similar. The block has two inputs and four outputs:
  - The upper input **1** is a signal that is internally added to the mean arrival rate parameter in order to calculate the instantaneous mean arrival rate parameter. This signal is useful when it is desired to test the controller in the face of traffic with varying parameters (section 3.9.2). The lower input **2** is an offset that is added internally to the mean service rate parameter to calculate the actual service rate. This signal is generated as the output of the controller.
  - The uppermost output **a** is the measured buffer quantile. The second output **b** is the weighted average of the measured and calculated buffer quantile (section 3.5.1). We refer to this output as the *plant output*. Output **c** is the number of cells arriving during the last controller sampling time (section 3.8.1). Output **d** is the current buffer occupancy in cells. Outputs **c** and **d** are used in dynamically adjusting the constraints (section 3.8.2).
- The block `pre` was originally responsible for smoothing the setpoint (reference signal)—to avoid large overshoot/undershoot—as follows:

$$\begin{aligned} y_i^* &= y_k & \text{if } i = k \\ &= \alpha\, y_{i-1}^* + (1-\alpha)\, y_i^{ref} & \text{if } i > k \end{aligned}$$

  where y_i^* is the smoothed setpoint at sampling instant i, y_k is the plant output at sampling instant k, $\alpha \in [0, 1)$ is a smoothing factor, and y_i^{ref} is the desired reference signal at time i.

We have modified `pre.m` to dynamically adjust the constraints on the service rate as well (section 3.8.2).

- The block `Controller` is responsible for recursively identifying a model (section 3.3.2.3 describes the RRLS algorithm used) of the queuing system using the sampled input and output. The order of the model is pre-specified. The block then computes a *minimum-bias* control effort (section 3.3.3) that satisfies the constraints specified by the `pre` block, based on the identified model. Multiple `Controller` blocks can be added in parallel, each with a different model order, resulting in the online identification of multiple models. We use a single controller block to speed up our simulations.

  The idea behind identification of multiple models is that, over the course of time, the plant may be most accurately represented by models of different orders. If during a particular interval, the plant output is fairly constant, then a model of small order may be able to capture the essential information. If during another interval, the plant output varies very rapidly, a higher model model may be necessary to capture the dynamics. Thus, if at each point in time, multiple models are obtained, then we can choose the control effort indicated by the most accurate model *at that point in time*. Alternatively, we can choose the weighted average of the control efforts specified by the different models, where the weights are based on the current accuracy of the models.

- The block `Selector` is responsible for identifying the good models at a given sampling instant. The poor models (large prediction error) are re-initialized. The control effort sent to the plant (queuing system) is the weighted average of the control efforts indicated by the good models. During an initial start up period, the control effort is a random signal generated by block `Random`.

3.8.1 Choosing the controller sampling time

In section 3.7.1, we mentioned that the output of the queuing system is sampled at a rate T_{sample} in order to estimate the buffer quantile using equation 3.9. In section 3.3.1, it was seen that the calculation of the discrete-time error signal $e(t)$ involved the sampling of the plant output and the reference signal. This may done at a sampling rate $T_{control\_sample}$ (which we term as the *controller* sampling rate to distinguish it from the *plant* sampling rate) which may be different from T_{sample}. The only requirement is that $T_{control\_sample} \geq T_{sample}$.

As an illustration, consider a controller sampling instant t_a. The next controller sample will occur at $t_b = t_a + T_{control\_sample}$. Between t_a and t_b, there can be at most $\eta = \lceil T_{control\_sample}/T_{sample} \rceil$ plant sampling instants. In each of these plant sampling intervals, as mentioned in section 3.5.1.2, the net arrival process is used to calculate the buffer quantile. The buffer quantile is measured at the sampling instants as well. The weighted average of the calculated and measured quantiles is used to form the plant output signal at the plant sampling instants and can be denoted by $y_i, i = 1, \cdots \eta$. To calculate the plant output $y(t)$ at the controller sampling instant t_b, one of the following methods can be employed:

- The calculated quantile at the last plant sampling instant, y_η, can be used as an estimate of $y(t)$. This has the benefit that the information used by the controller is as upto-date as possible.

- The average of the past η samples, $y_1, \cdots, y_\eta$, can be used to estimate $y(t)$. The advantage of this is that the controller sample is less affected by random fluctuation in the last plant sample.

- As a compromise between the above two approaches, we can use the average computed over a window of the last w samples, where the window $w \in [1, \eta]$. A choice of $w = 1$ leads to the first approach, while $w = \eta$ gives us the second.

The choice of $T_{control\_sample}$ is based on the following considerations:

- A high rate of sampling enables the controller to react to deviations in the plant output from the desired value as quickly as possible. Too high a sampling rate, however, may result in instability of the closed loop system[11], and may pose an unacceptable computational overhead as well. Finally, our analysis in section 3.3.2.2 assumed a system delay d of one sampling period. This means that the identification routine must be able to detect a response to a stimulus after one sampling period. If the system is slow to respond, the sampling period must be decreased accordingly so that there is appreciable change in the output after one sampling interval. This places an upper limit on the sampling rate.
- Too low a sampling rate may result in poor identification due to loss of valuable information. Also, the controller may not be able to react quickly enough to disturbances.

We found that although the measured quantile reacts very sluggishly to a change in the service rate, the calculated quantile responds quite quickly and therefore the choice of unit system delay is not too restrictive. After some experiments using Poisson traffic with the characteristics described in section 3.9.1.1, we chose $T_{control\_sample} = 1$ second ($\eta = 10$). As sections 3.9.1.2 and 3.9.1.3 show, this choice is general enough that both batch Poisson and self-similar traffic could be well controlled.

3.8.2 Service rate constraints

The following discussion is with reference to figure 3.15. The service rate is 150 cells/sec and the mean arrival rate is 120 cells/sec. The reference level is 35 cells as before. CAC specifies an upper and lower limit on the service rate in terms of the

---

[11] As the sampling rate increases, the discrete-time poles and zeros move outward from the origin towards the unit circle. Thus a high sampling rate may lead to either a non minimum-phase system (zeros outside the unit circle) that is closed-loop unstable or to an unstable plant (poles outside the unit circle). Depending on the controller, one or both or neither of the above problems may be prevented from destabilizing the control system.

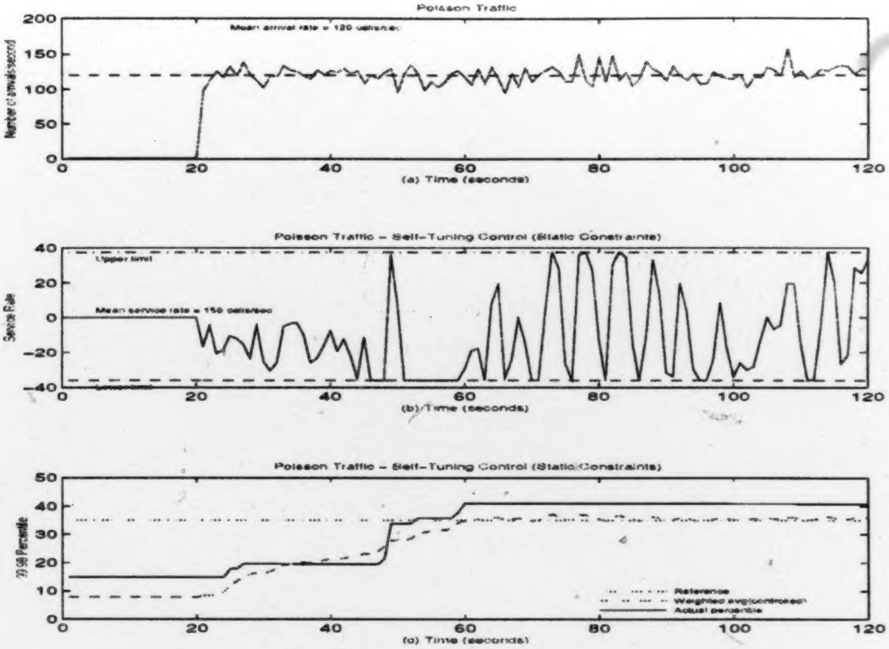


Figure 3.15: Feedback control - static constraints

allowable positive and negative deviation, respectively, from the mean service rate. The controller performance is shown in figure (c) while the service rate indicated by the controller is shown in figure (b). Once again, the control based on the calculated quantile can be seen to be accurate.

Considering the portion of the figures (b) and (c) in the time period between 50 and 55 seconds, we note the following :

- In figure (c), the calculated quantile is very close to the reference value. However, the lower limit for the service rate (figure (b)) is still -38.
- The calculated quantile is constant for a short while during the period under consideration. The controller therefore finds no change in the plant output as the service rate is lowered. In an effort to take the plant output to the reference value, the service rate is lowered till it reaches the lower limit (figure(b)). At this point the service rate is very close to the arrival rate and it is possible for

a large overshoot to occur—this is the reason for the overshoot roughly at time $t = 60$ seconds.

When an overshoot occurs (figure (c)), the plant output decreases very slowly towards the reference value. For our queuing system, this means that QoS will be violated for a long time. This is a consequence of the small gain for positive step inputs mentioned in section 3.7. Therefore, we should be more cautious in reducing the service rate as the plant output nears the reference value. The magnitude of the lower limit on the service rate should be progressively decreased as the plant output approaches the reference value. This argues for the use of *dynamic* constraints instead of *static* constraints. We now present a heuristic scheme for varying the constraints dynamically from the initial values specified by CAC. The following symbols will be used in the description of the algorithm:

y_t The measured buffer quantile at time t.

y_t^* The plant output (weighted average) at time t.

y^{ref} The reference level (assumed to be constant).

y_t^{ref} The smoothed reference level at time t (section 3.8).

v_t^* The desired rate of approach of plant output to the reference level, at time t.

v_t The actual rate of approach of plant output to the reference level, at time t.

λ_t The mean arrival rate measured up to time t.

σ_t The standard deviation of the arrival rate measured up to time t.

μ The mean service rate—specified by the CAC.

μ_{max}	The **static** upper limit on the service rate—specified by the CAC as the maximum deviation from μ in the positive direction.
μ_{min}	The **static** lower limit on the service rate—specified by the CAC as the maximum deviation from μ in the negative direction.
u_t^{min}	The (**dynamic**) lower limit of service rate variation (< 0) at time t.
u_t^{max}	The (**dynamic**) upper limit of service rate variation (> 0) at time t.
u_t	The plant input (controller output) at time t, specified as a deviation from μ and constrained to lie in $[u_t^{min}, u_t^{max}]$. The actual service rate used in the queuing system is $\mu + u_t$.
κ	A constant (> 2) that controls the rate of decrease of u_t^{min}.
K_t	The buffer occupancy at time t.

At any time t, a smooth trajectory between y_t and y^{ref} is calculated (section 3.8). The first point in the trajectory is the reference value, y_t^{ref}, for the controller at time t. Therefore, y_t^{ref} represents an intermediate goal—the controller will try to take the plant output to y_t^{ref} in the next sample period. It should be noted that an increase in the service rate results in a decrease in the plant output and *vice versa.* This means that

$$y_t > y_t^{ref} \Longrightarrow u_t > 0, \quad y_t < y_t^{ref} \Longrightarrow u_t < 0$$

The lower limit u_t^{min} is obtained as follows:

$$u_t^{min} = \begin{cases} \min(u_t^1, u_t^2, u_t^3) & \text{if } y_t < y^{ref} \\ -1 & \text{if } y_t \geq y^{ref} \end{cases} \tag{3.11}$$

The constraints u_t^1, u_t^2 and u_t^3 are described below. When the plant output is less than the reference value, i.e. $y_t < y^{ref}$, the first step is to to determine the rate of approach, v_t, of y_t towards the intermediate goal y_t^{ref}:

$$v_t = y_t - y_{t-1}$$

Since we would like the controller to take the plant output to y_t^{ref} in the next time step, the desired rate of approach is

$$v_t^* = y_t^{ref} - y_t$$

The plant output is said to approach the reference *slowly* if the following condition holds:

$$v_t < v_t^* \text{ or } v_t < 0$$

Otherwise the rate of approach is taken to be *fast*.

The idea behind the constraint u_t^1 is the following: when the plant output approaches the reference from below ($y_t^* < y^{ref}$) slowly, we would like to reduce the service rate further. This means that we need to increase $\mid u_t^{min} \mid$ to allow a greater decrease in the service rate.

Thus,

$$u_t^1 = a\, u_{t-1}^{min}$$

where $a = 1 - y_t/y^{ref}$ measures the relative error of the plant output with respect to the reference value. When $y_t < y^{ref}$, $a \in [1, 2]$. This means that we increase $\mid u_t^{min} \mid$ cautiously when y_t is close to y^{ref} (small a) and increase it rapidly when the relative error is large.

However, if the output is approaching the reference rapidly from below, we need to increase the service rate and thereby slow down the rate of approach. This service rate increase is achieved by reducing $\mid u_t^{min} \mid$, automatically forcing the service rate to increase. Therefore, when the rate of approach is too fast,

$$u_t^{min} = u_{t-1}^{min}/\kappa$$

That is, it is proposed to reduce $\mid u_t^{min} \mid$ by a constant factor. This geometric decrease over a number of time steps makes it possible to rapidly increase the service rate to μ when the plant output is fast approaching the reference. Because $\kappa > a$ always, u_t^1 will always decrease at a faster rate than it increases. This is a necessary condition for the mechanism to be stable. This mechanism is similar to the *additive increase and multiplicative decrease* congestion control mechanism in TCP ([28]).

The lower limit u_t^2 must satisfy the following condition:

$$| u_t^2 | < \mu_{mean} - \lambda_t$$

since the mean service rate cannot fall below the mean arrival rate for long periods of time if the queue is to be stable.

The choice of u_t^3 is based on the following information:

- The current buffer occupancy K_t
- The mean (λ_t) and standard deviation (σ_t) of the arrival process, measured up to time t.

The arrival process is assumed to be normally distributed (section 3.5.1.2) with parameters $< m, \sigma >$ estimated at time t by $< \lambda_t, \sigma_t >$. Therefore, using a table of the standard normal distribution, the maximum number of arrivals, L, in a sample period is given by

$$L \leq \lambda_t + 3\ \sigma_t, \text{ with } 99.9\% \text{ probability}$$

Lowering the service rate below the mean value μ has the risk that a large number of arrivals in the next sampling period may drive the plant output above the reference value. If the service rate is lowered to a value u_t^3, then the risk is $\lambda_t + 3\ \sigma_t - u_t^3$. Since the service rate cannot be lowered below the mean arrival rate (estimated by λ_t), the maximum risk is $\lambda_t + 3\ \sigma_t - \lambda_t = 3\sigma_t$. Another choice for the risk factor is the difference b between the reference level y^{ref} and the current buffer occupancy K_t:

$$b = y^{ref} - K_t$$

When b is large, it is permissible to allow a greater decrease in the service rate, i.e to take a greater risk. The possible risk $risk'$ is chosen as the minimum of the two factors

$$risk' = min\,(3\ \sigma_t, b)$$

The actual risk, $risk$, is obtained by after adding $risk'$ to the difference between the mean service rate and the maximum probable number of arrivals in a sampling period and then weighting the sum by a factor $c, c \in (0, 1]$

$$risk = c\,(\mu - L + risk')$$

The factor c is chosen to be at its maximum value of 1 when the relative error between the plant output and the reference value, a, is greater than 20% [12]. In that case, the actual risk, $risk = risk'$, meaning that we are willing to take the entire risk as the output is far away from the reference. However, when the output is close to the reference we are not willing to take all the risk. Hence, when the relative error is less than 20%, the weighting factor decreases rapidly along a straight line. When the relative error is less than 10%, the c levels off at a value of 0.1. c can be formally defined as

$$c = \begin{cases} 1 & \text{if } a > 0.2 \\ 9\,a - 0.8 & \text{if } a \in [0.1, 0.2] \\ 0.1 & \text{if } a < 0.1 \end{cases}$$

Putting the above statements together, the constraint u_t^3 is given by

$$u_t^3 = min\ (0, risk)$$

The upper limit u_t^{max} is chosen as

$$u_t^{max} = \mu^{max}$$

3.9 Results

In this section, we study the performance of the controller under varying circumstances. Section 3.9.1 describes the control of the queuing system with traffic generated by the Poisson, batch Poisson and self-similar models described in section 3.6. The average arrival rate of the traffic is assumed to be constant. The goal is to study the ability of the controller to maintain the output of the system at a steady reference value.

---

[12] The assumption is made that a relative error in the plant output of up to 20% is acceptable

Next, we allow the average arrival rate to vary during the course of the simulation. This will cause the queue utilization ρ to vary as well. As figure 3.11 shows, this will in turn cause the plant gain ($\frac{\Delta y}{\Delta u}$) to vary, requiring the controller to adapt to the new plant characteristics. The performance of the adaptive controller is described in section 3.9.2. Section 3.9.3 studies the effect of the forgetting factor (section 3.3.2.3) while section 3.9.4 describes the behavior of the system for different bandwidths allocated by CAC.

3.9.1 Self-tuning controller

The queuing system is fed with bursty traffic, the average arrival rate (SCR) of which is constant. The randomness of the cell inter-arrival times produces a stochastic disturbance of the system output. In this section, we will study the ability of the controller to take the plant output to the reference level in spite of the disturbance. It is usual to demonstrate the ability of the controller to track changes in setpoint accurately. In the system under consideration, the reference signal is a measure of the required quality of service. The setpoint is likely[13] to be constant over a period that is large compared to the time required by the controller to achieve satisfactory control. For this reason, we will only study the *regulator* problem where the disturbance signal is large and the reference signal is constant.

For each traffic model (Poisson, batch Poisson, self-similar), the mean arrival rate SCR is fixed at 120 cells/sec. Section 3.9.1 explains why this unrealistic value was chosen, and how realistic arrival and service rates can be handled. The system is studied at medium to high utilizations: $\rho = 0.6, 0.7, 0.8$, corresponding to initial bandwidth allocations by CAC of $\mu = 150, 170, 200$ cells/sec. Each experiment is repeated with three different seeds for the random number generator, resulting in a total of nine experiments per traffic type.

---

[13] QoS re-negotiations may cause the setpoint to vary, albeit very slowly in comparison to the controller sampling period

The upper bound on the service rate, μ_{max}, is assumed to be 25% of the mean service rate μ. The lower bound μ_{min} is given by

$$\mu_{min} = \mu - 1.01\ \text{SCR}$$

As a result of this constraint, the instantaneous service rate can never fall below SCR. In each experiment, the system is allowed to run in open-loop mode for 20 seconds. The excitation to the system is zero during this time. For the next 20 seconds, the service rate is varied randomly (within the specified bounds). The system input-output data is stored for use by the identification routine. The controller starts up around $t = 45$ seconds.

The queuing system is sampled every 0.1 seconds, i.e. $T_{sample} = 0.1$, in order to estimate the buffer quantile. The controller sampling time, $T_{control\_sample}$, is 1 second. In order to obtain the sampling intervals for processes with other time-scales, the unit of time can be selected so that the absolute value of the mean of the net arrival process, $|\ \mu\ |$ is of the order of a few cells. For example, if the mean arrival rate of the traffic is 100000 cells/sec (42.4 Mb/s), and the mean service rate is 120000 cells/sec, then $T_{sample} = 0.25$ ms results in $|\ \mu\ | = 5$ cells. $T_{control\_sample} = 2.5$ ms implies that up to ten samples of the plant output can be aggregated to form the controller input at the controller sampling instant ($\eta = 10$).

This reasoning also shows why we chose traffic with an unrealistic value for SCR (120 cells/sec or 50Kb/sec). This arrival rate allowed us to choose the sampling interval as one second for simplicity. At the same time, traffic with higher arrival rates can be handled by simply rescaling the sampling intervals as shown above. In addition, we can see that the sampling intervals are dependent only on the time scale of the net arrival process, unlike the scheme described in [29] where the sampling rate is limited by the propagation delay between ATM switches.

3.9.1.1 Poisson traffic

Figures 3.16(c), 3.17(c) and 3.18(c) show the controller performance for $\rho = 0.8$. The reference level was chosen to be 35 cells. At the end of the simulation, the actual buffer quantile is close to the reference level in all three cases. Sub-figure (b) in each of the three figures shows the instantaneous service rate as well as the upper and lower constraints. We can observe the variation of the lower constraint according to the algorithm outlined in section 3.8.2—in particular, the lower constraint almost reduces to zero when the actual quantile is close to the reference level. Sub-figure (a) shows the mean number of arrivals per sampling period. Since the controller sampling period is 1 second, this is equivalent to the mean arrival rate in cells/sec.

In figure 3.17 (sub-figures (b) and (c)), we can see evidence of the *additive increase and multiplicative decrease* mechanism. At around $t = 170$ seconds, the buffer quantile increases, causing the lower constraint to rise rapidly to prevent an overshoot. Once the danger is past, the lower constraint decreases gradually, with small occasional jumps, to permit the quantile to rise closer to the reference.

Figures 3.19(c), 3.20(c) and 3.21(c) show the controller performance for $\rho = 0.7$. The error in all three cases is less than 15%. The three figures differ considerably (as in the previous case), demonstrating the value of testing the controller performance using more than one random number generator seed.

Figures 3.22(c), 3.23(c) and 3.24(c) show the controller performance for $\rho = 0.6$. Once again, the error in all three cases is less than 15% even though there is a small overshoot in figure 3.24(c). During the period that the weighted quantile (the system output) exceeds the reference value, the controller correctly increases the instantaneous service rate to the maximum possible value to reduce the buffer quantile. However, once the system output falls below the reference value, the service rate is reduced to μ_t^{min} (figure 3.24 b).

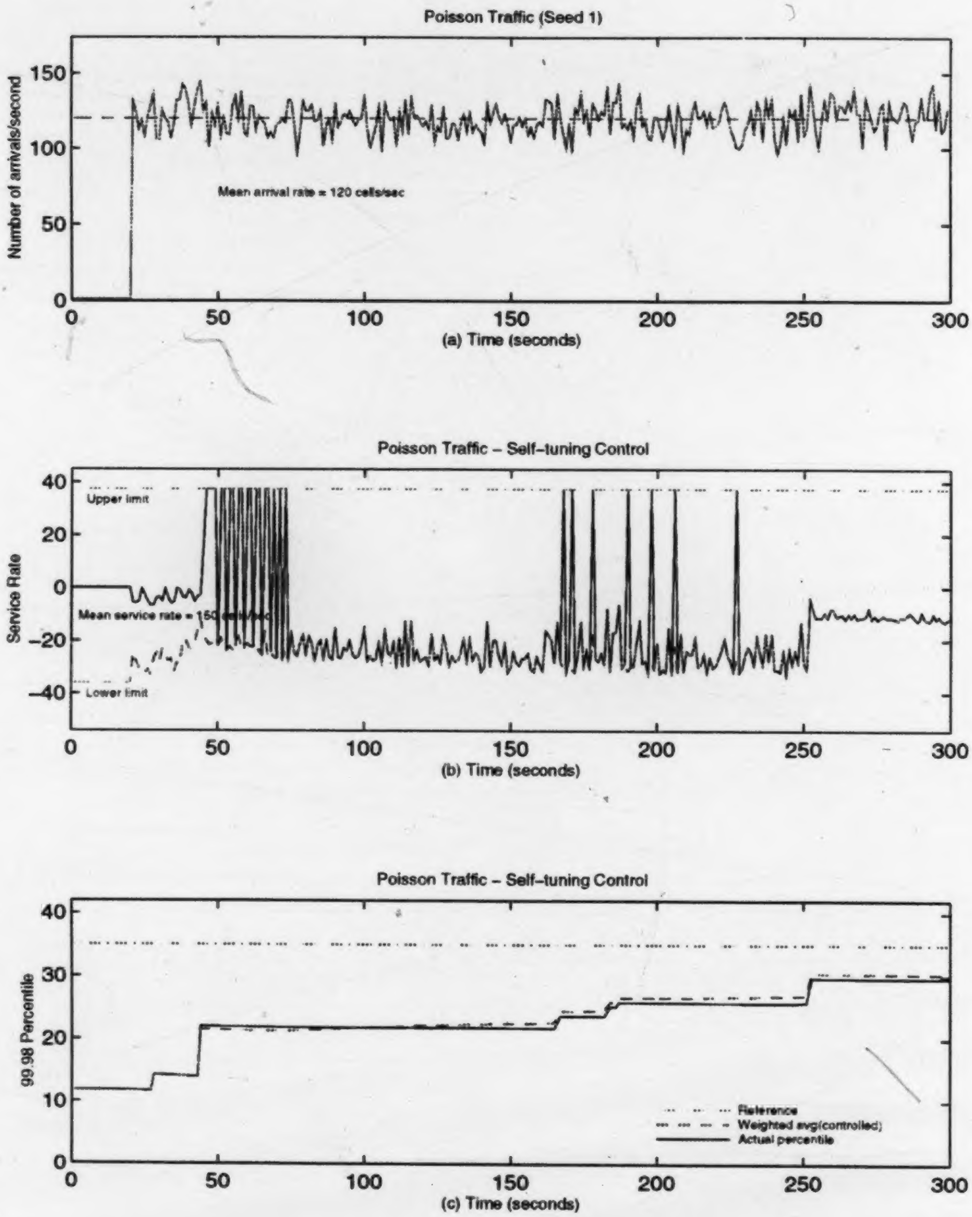


Figure 3.16: Self-tuning control (Poisson traffic, $\rho = 0.8$, seed = 1)

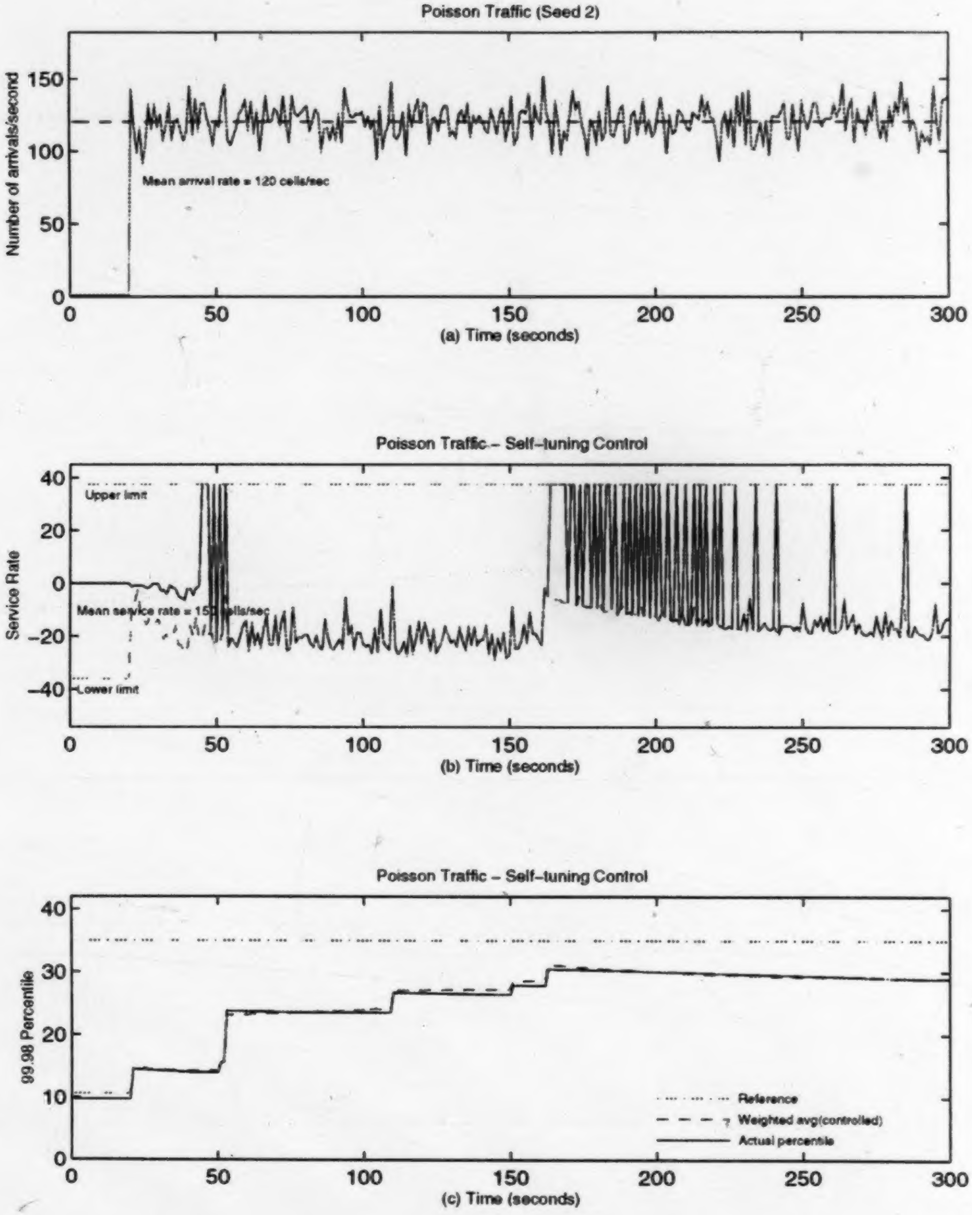


Figure 3.17: Self-tuning control (Poisson traffic, $\rho = 0.8$, seed = 2)

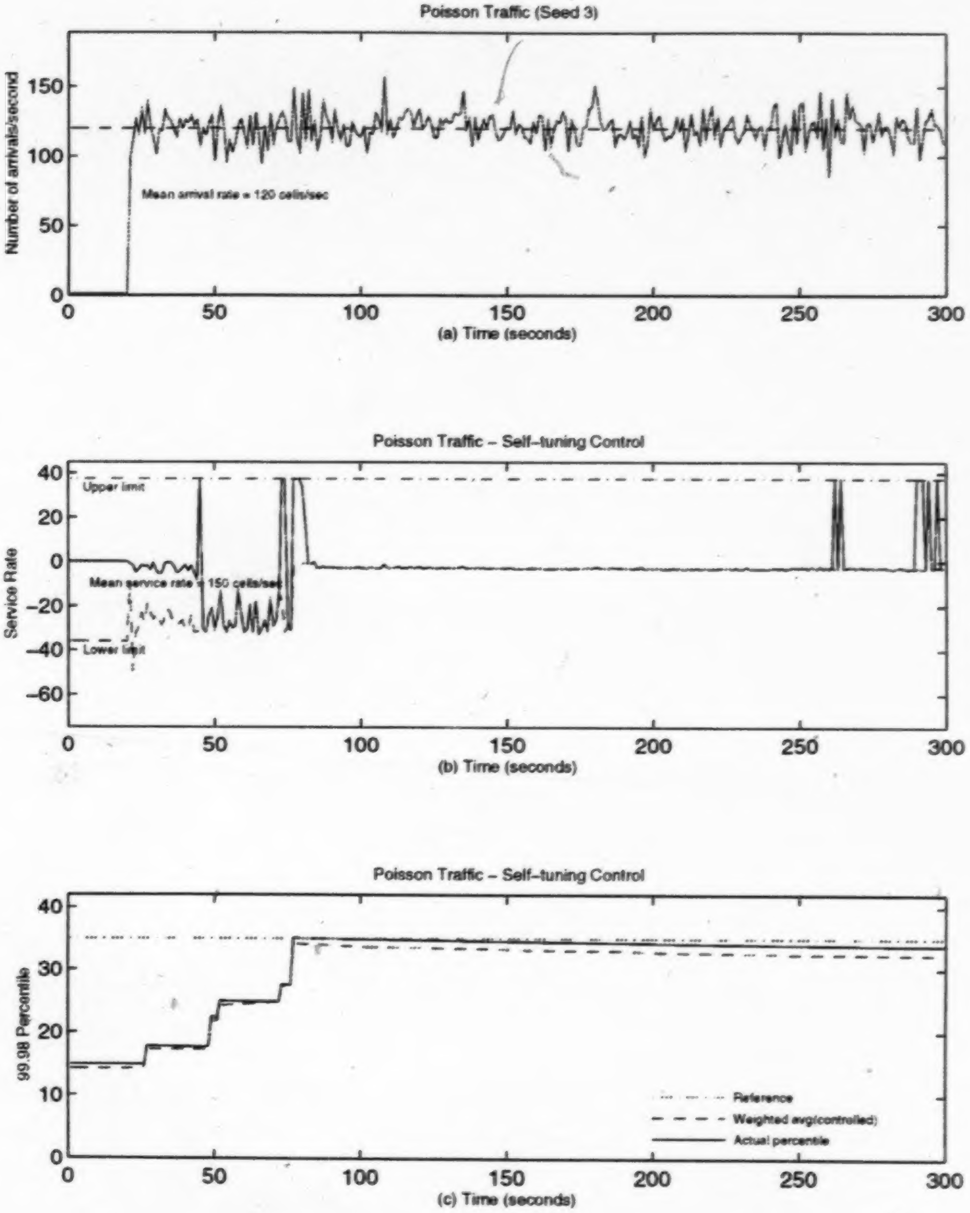


Figure 3.18: Self-tuning control (Poisson traffic, $\rho = 0.8$, seed = 3)

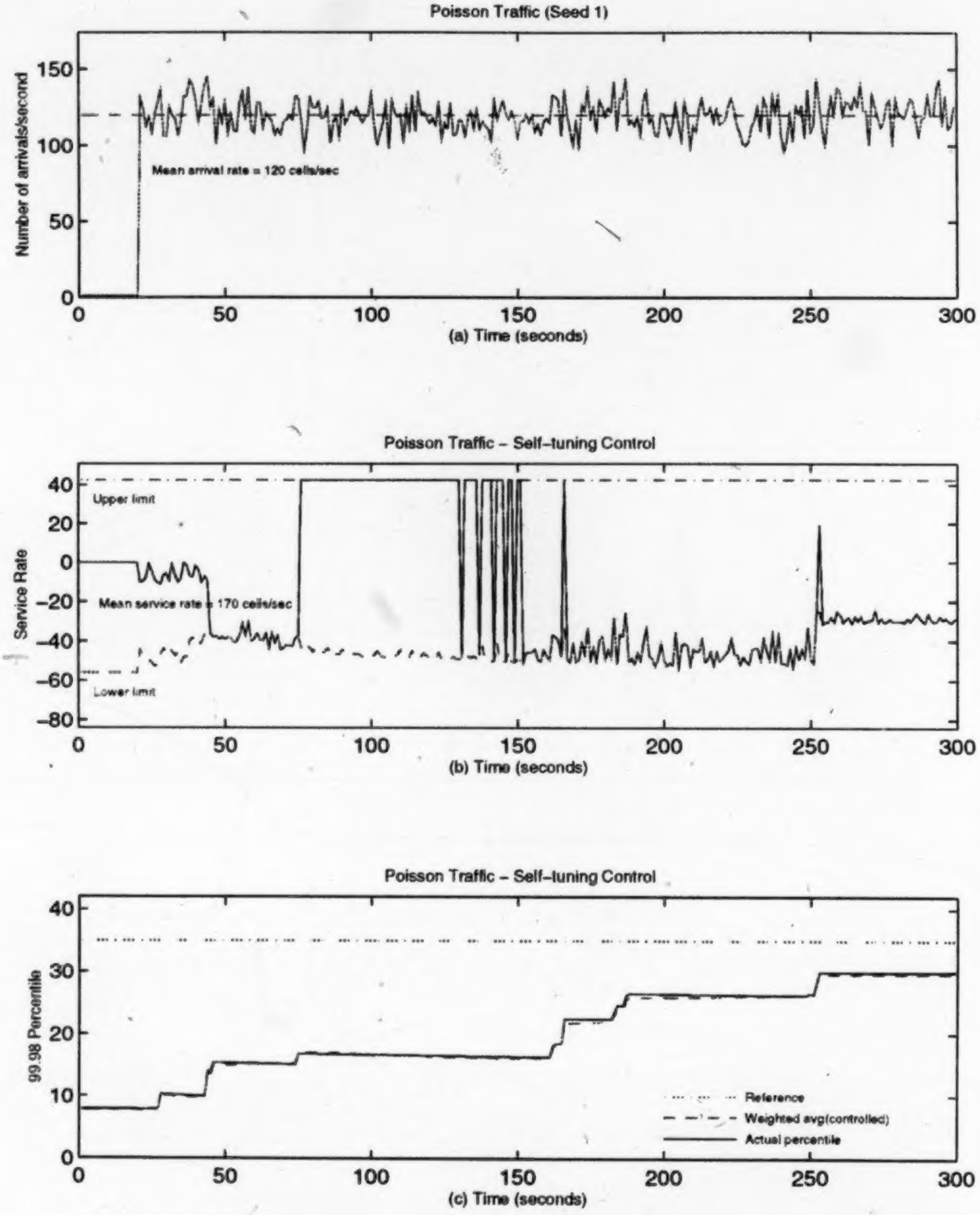


Figure 3.19: Self-tuning control (Poisson traffic, $\rho = 0.75$, seed = 1)

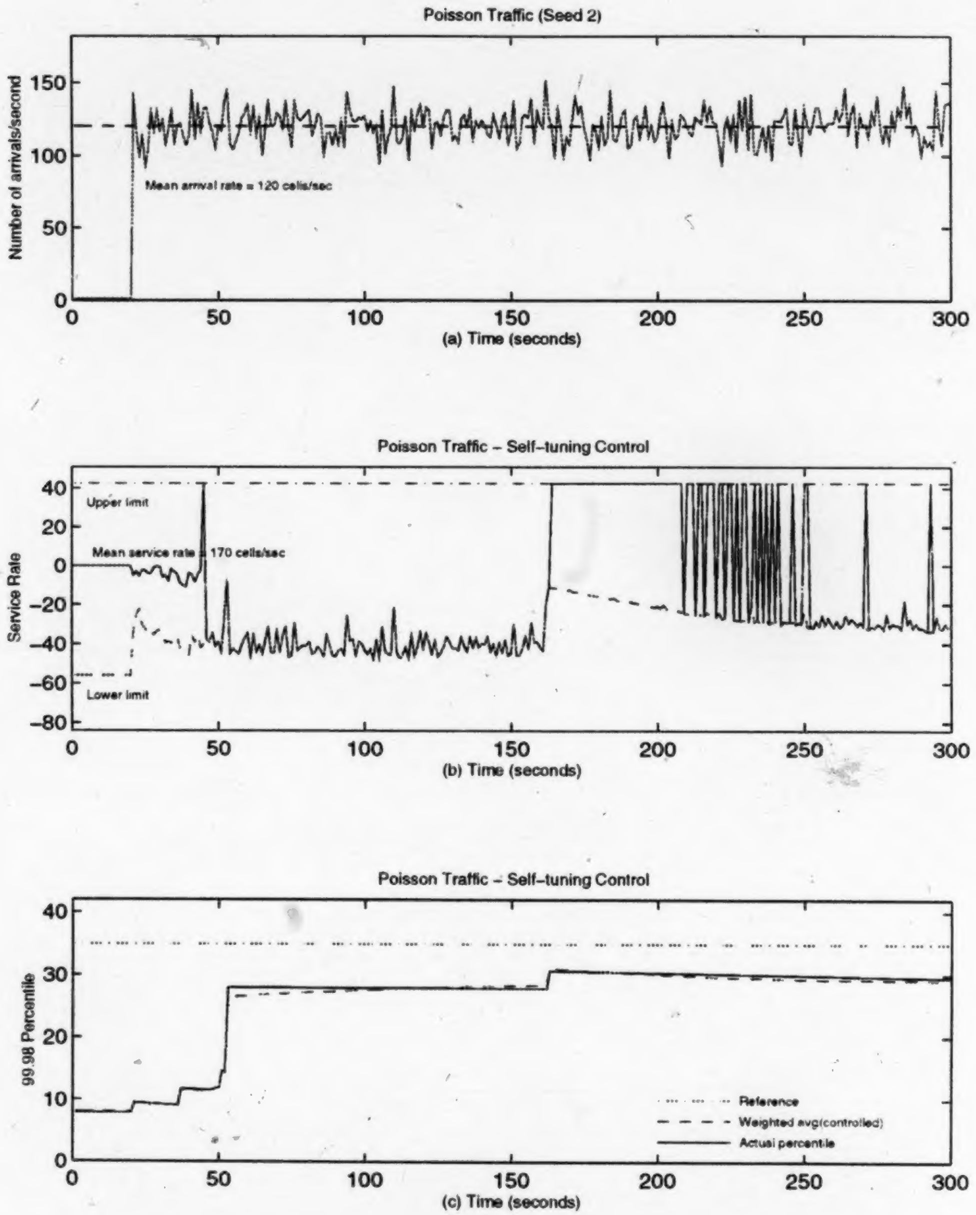


Figure 3.20: Self-tuning control (Poisson traffic, $\rho = 0.75$, seed = 2)

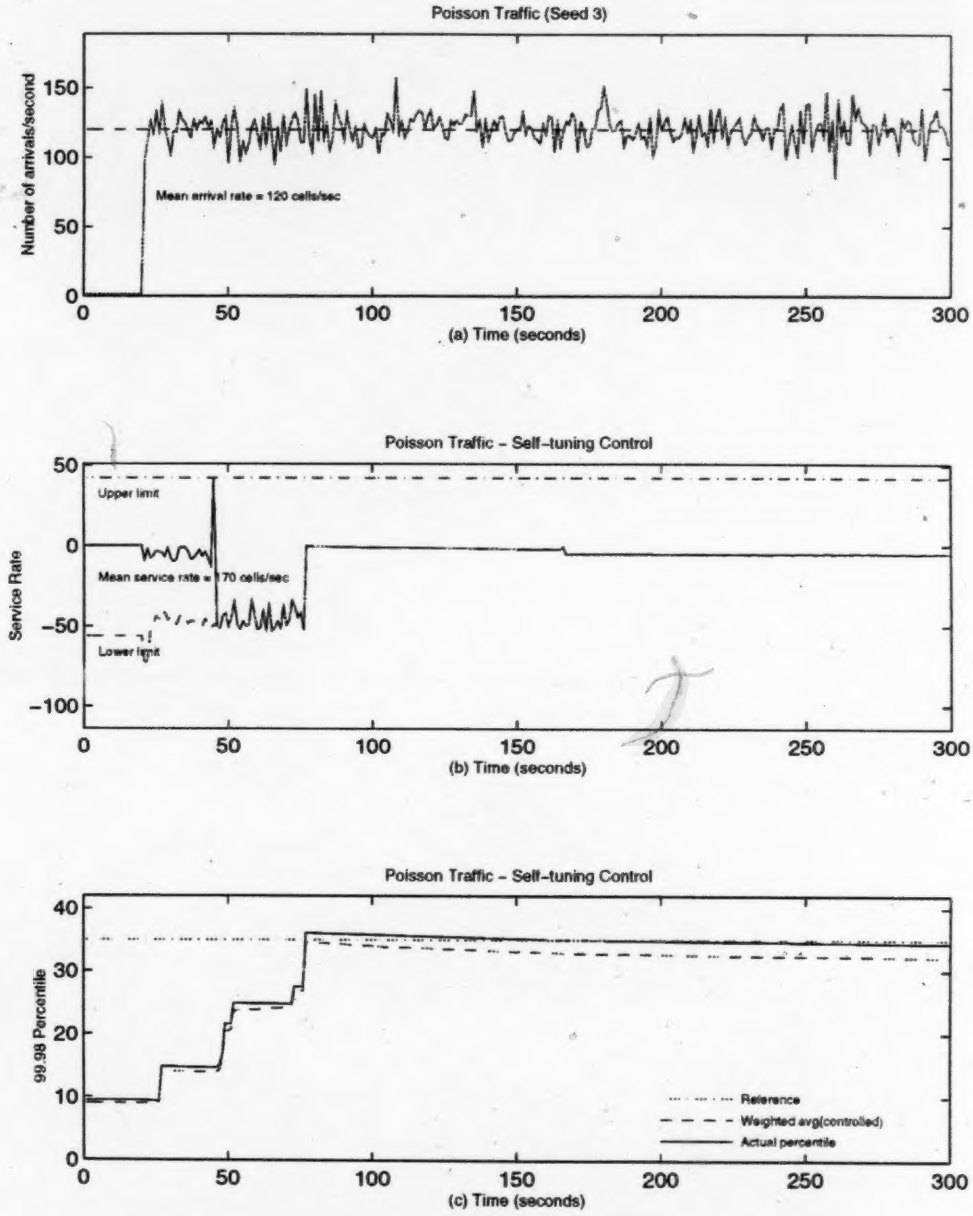


Figure 3.21: Self-tuning control (Poisson traffic, $\rho = 0.75$, seed = 3)

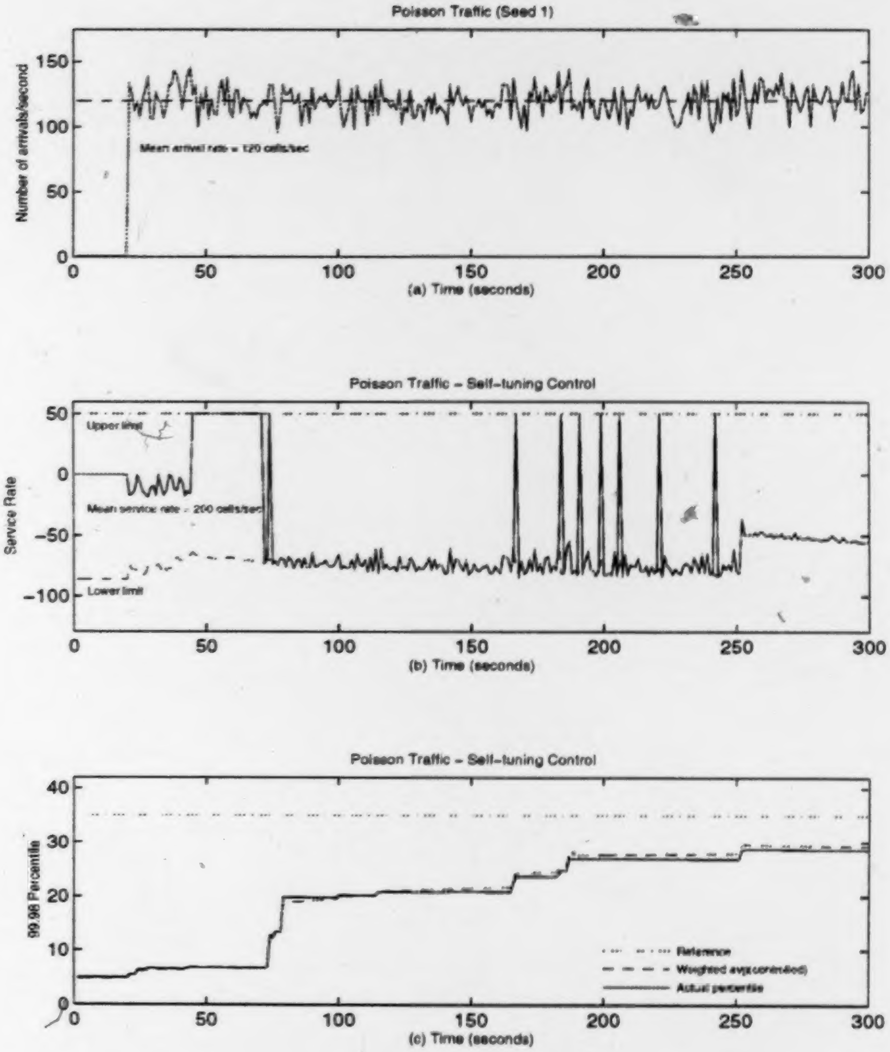


Figure 3.22: Self-tuning control (Poisson traffic, $\rho = 0.6$, seed = 1)

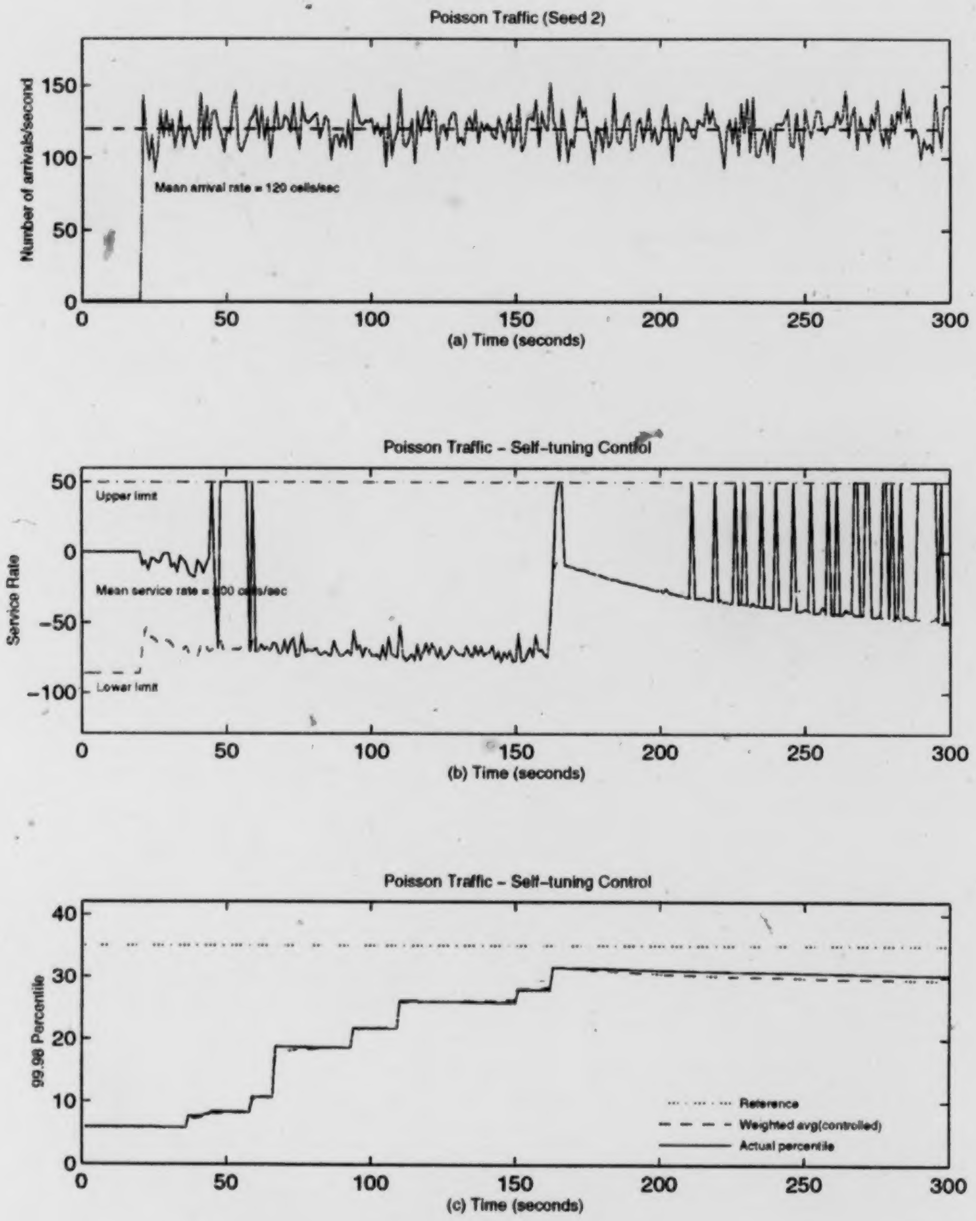


Figure 3.23: Self-tuning control (Poisson traffic, $\rho = 0.6$, seed = 2)

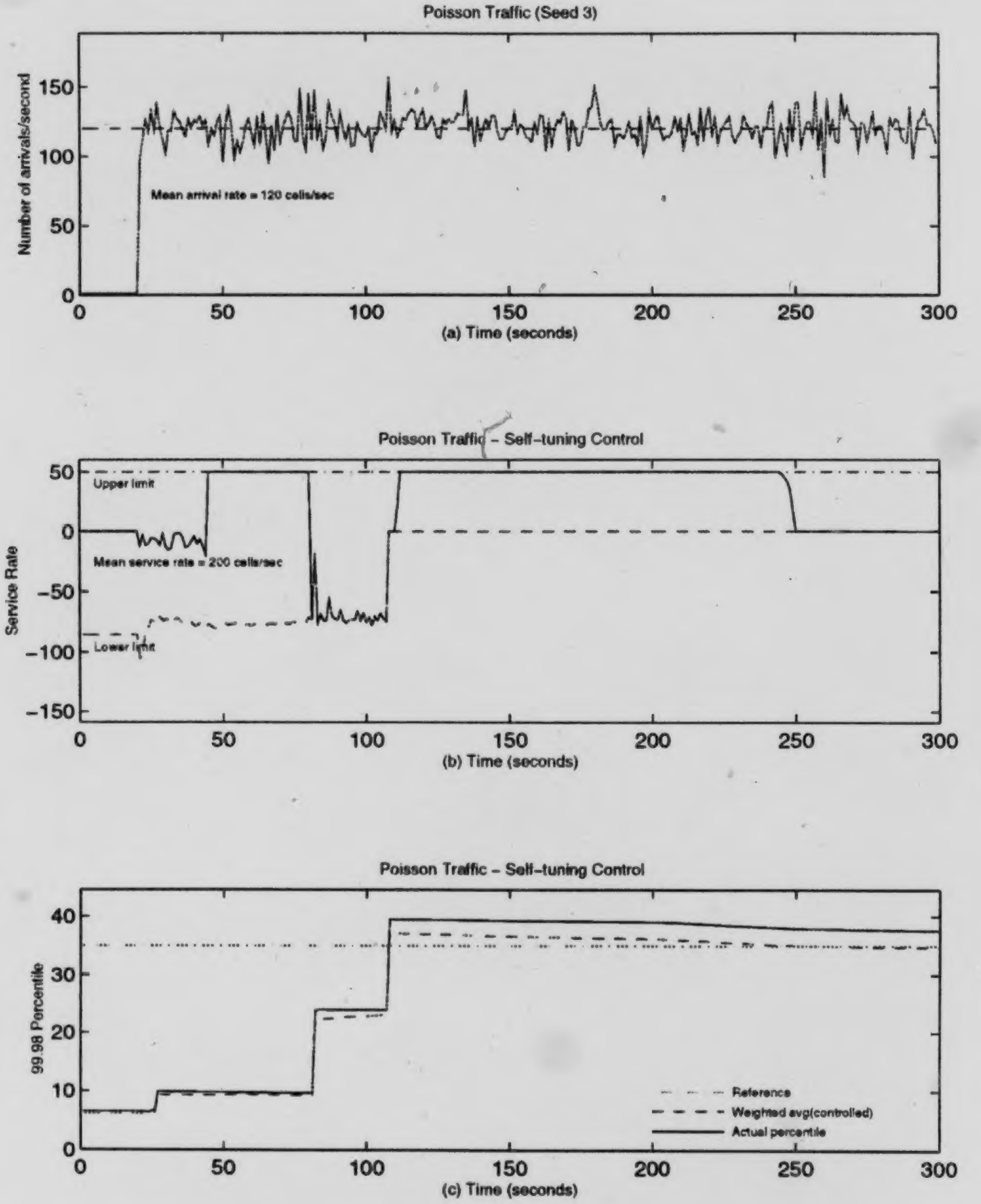


Figure 3.24: Self-tuning control (Poisson traffic, $\rho = 0.6$, seed = 3)

3.9.1.2 Batch-Poisson traffic

The batch Poisson traffic has the following parameters:

- The mean number of cells per arrival epoch, L, is chosen to be 5.
- The mean arrival rate, SCR, must be 120 cells/sec. Therefore, the epoch rate $\lambda = 120/5 = 24$/second.
- The QoS reference level is 150 cells.

Figures 3.25(c), 3.26(c) and 3.27(c) show the controller performance for $\rho = 0.8$. The relative error is less than 15% as in the case of Poisson traffic. Sub-figure (a) shows the mean arrival rate in each case. The traffic can be seen to be much more bursty than in the case of Poisson traffic.

Figures 3.28(c), 3.29(c) and 3.30(c) show the controller performance for $\rho = 0.7$. Once again, the error in all three cases is less than 15%. However, a small overshoot can be seen in figure 3.28(c). Though the controller does not immediately increase the instantaneous service rate to the maximum possible value when the calculated quantile (the system output) exceeds the reference value (around $t = 175$ seconds), we can see that the service rate is indeed increased at about $t = 275$ seconds (figure 3.28(b)).

Figures 3.31(c), 3.32(c) and 3.33(c) show the controller performance for $\rho = 0.6$. The controller performance can be seen to be fast and quite accurate.

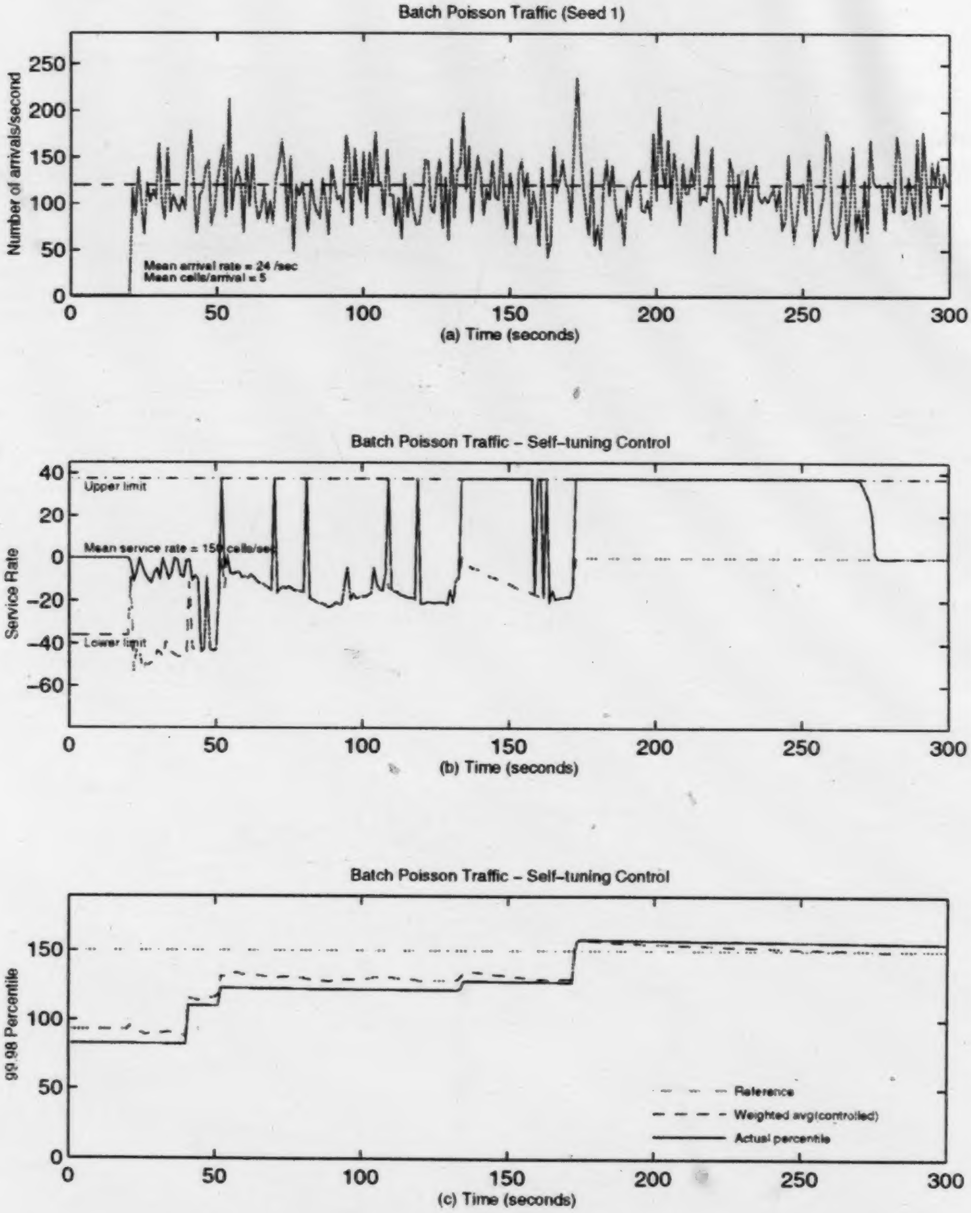
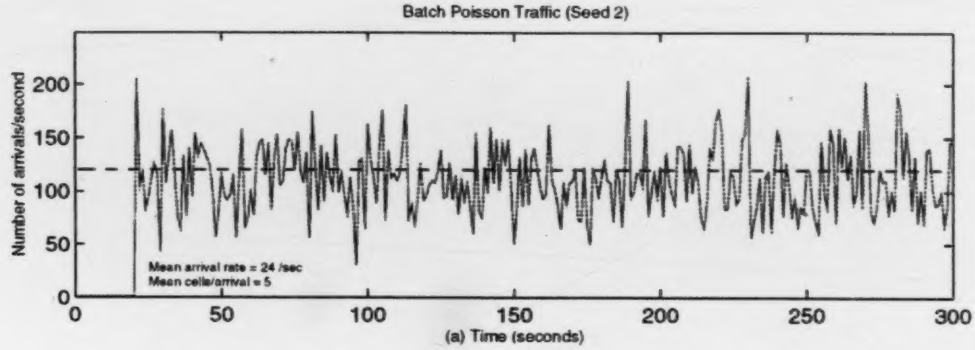
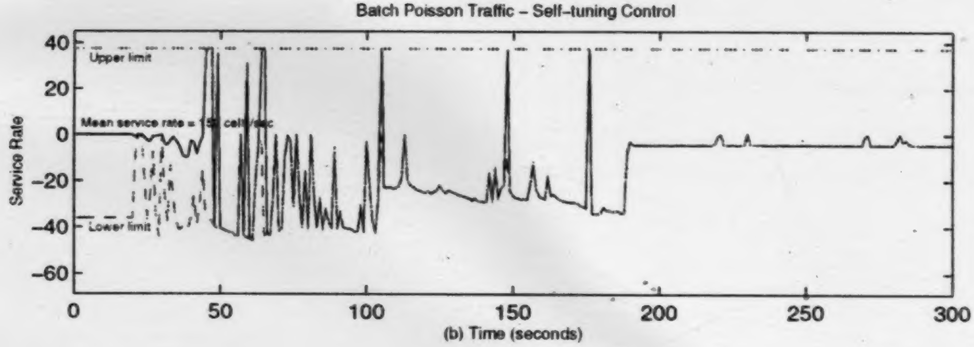


Figure 3.25: Self-tuning control (Batch Poisson traffic, $\rho = 0.8$, seed = 1)





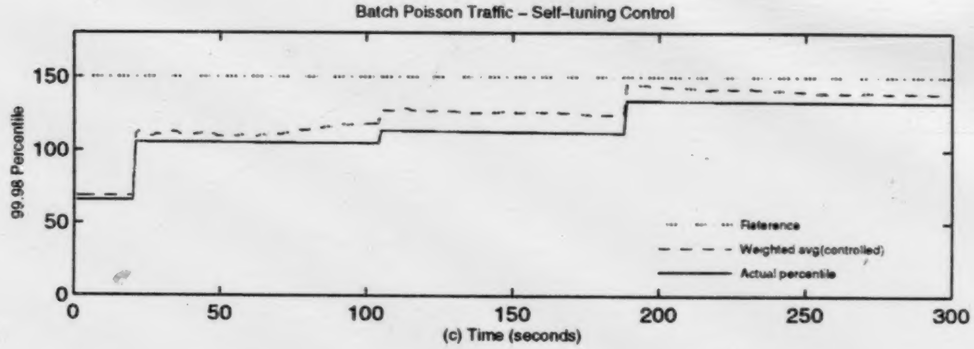


Figure 3.26: Self-tuning control (Batch Poisson traffic, $\rho = 0.8$, seed = 2)

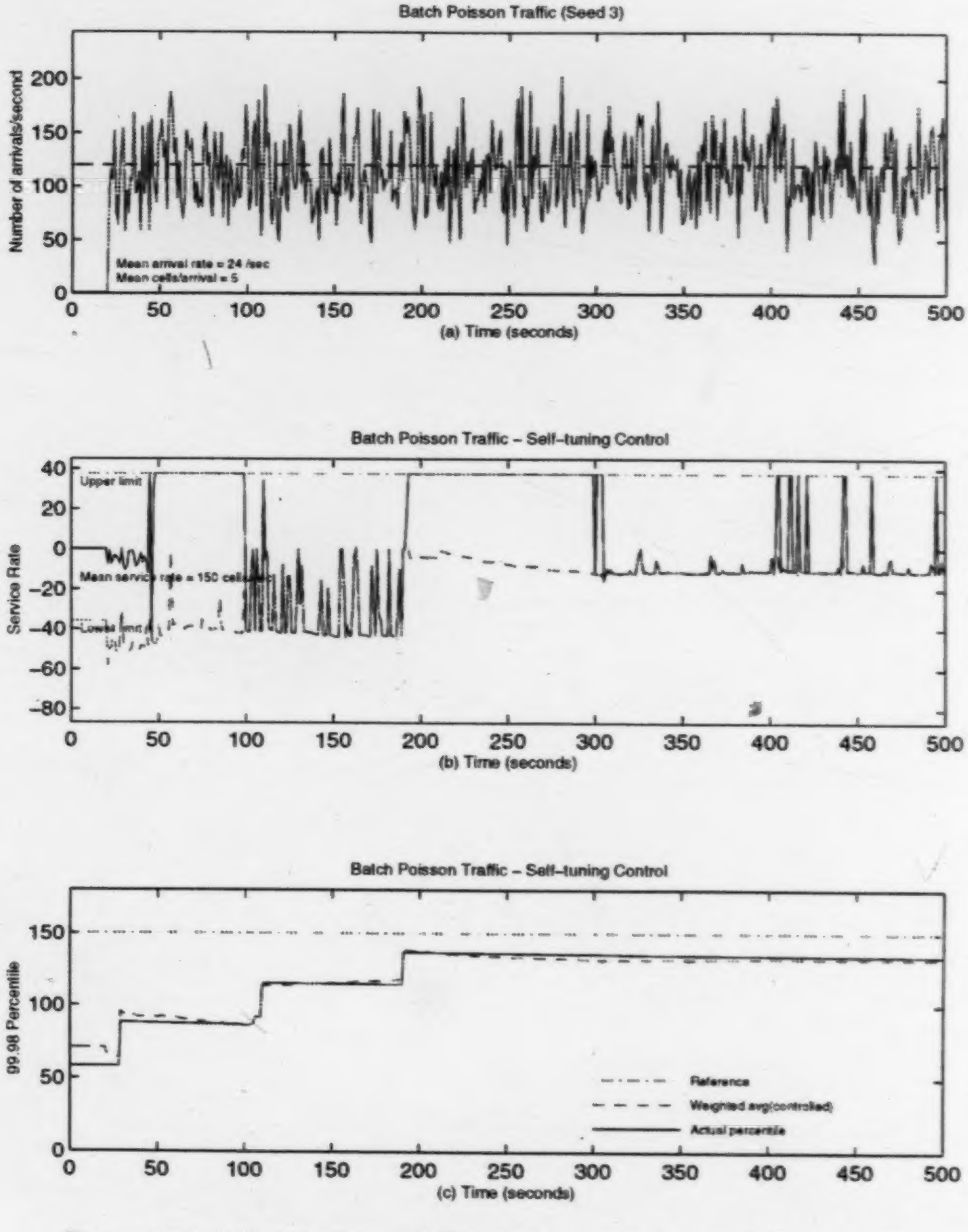


Figure 3.27: Self-tuning control (Batch Poisson traffic, $\rho = 0.8$, seed = 3)

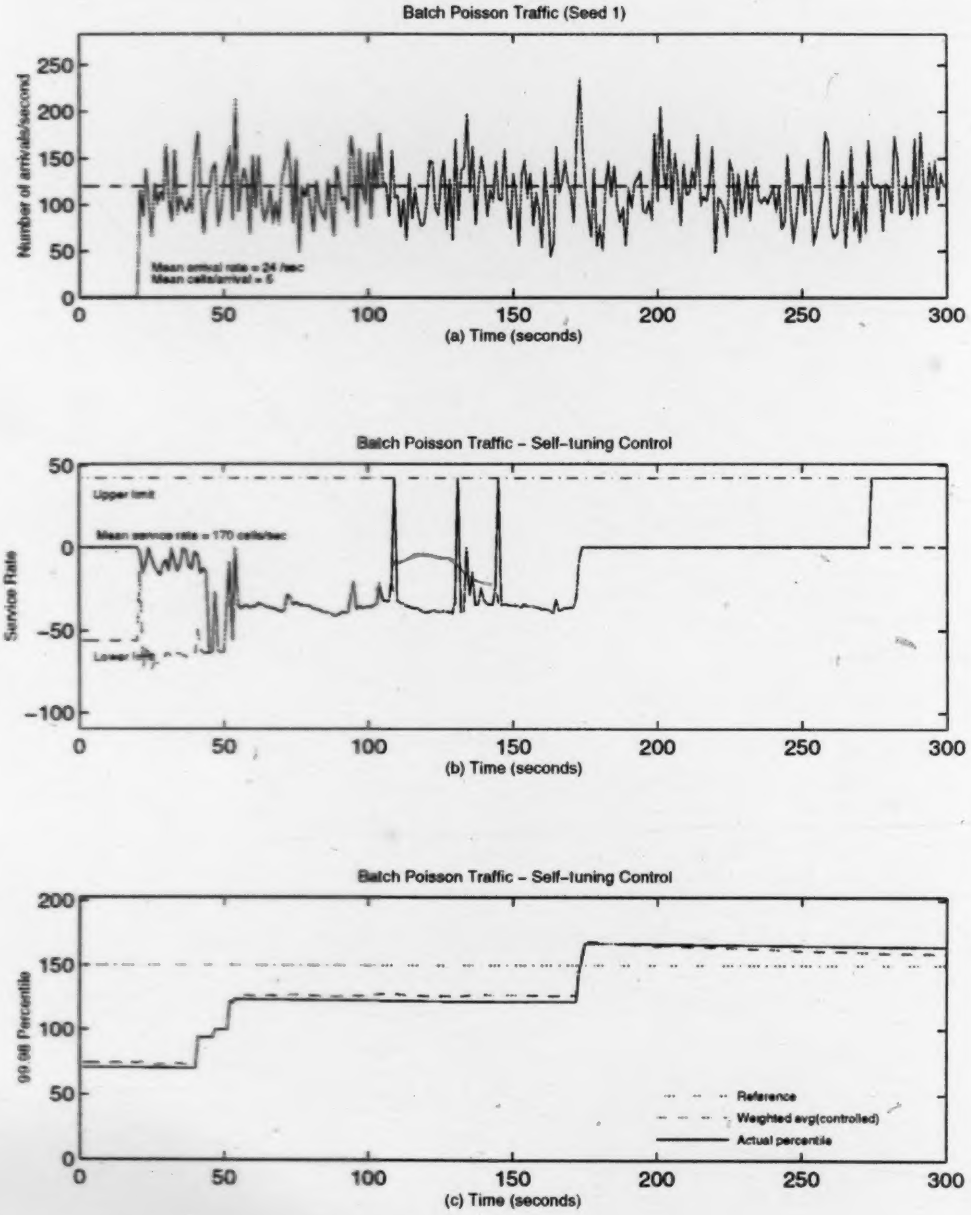


Figure 3.28: Self-tuning control (Batch Poisson traffic, $\rho = 0.75$, seed = 1)

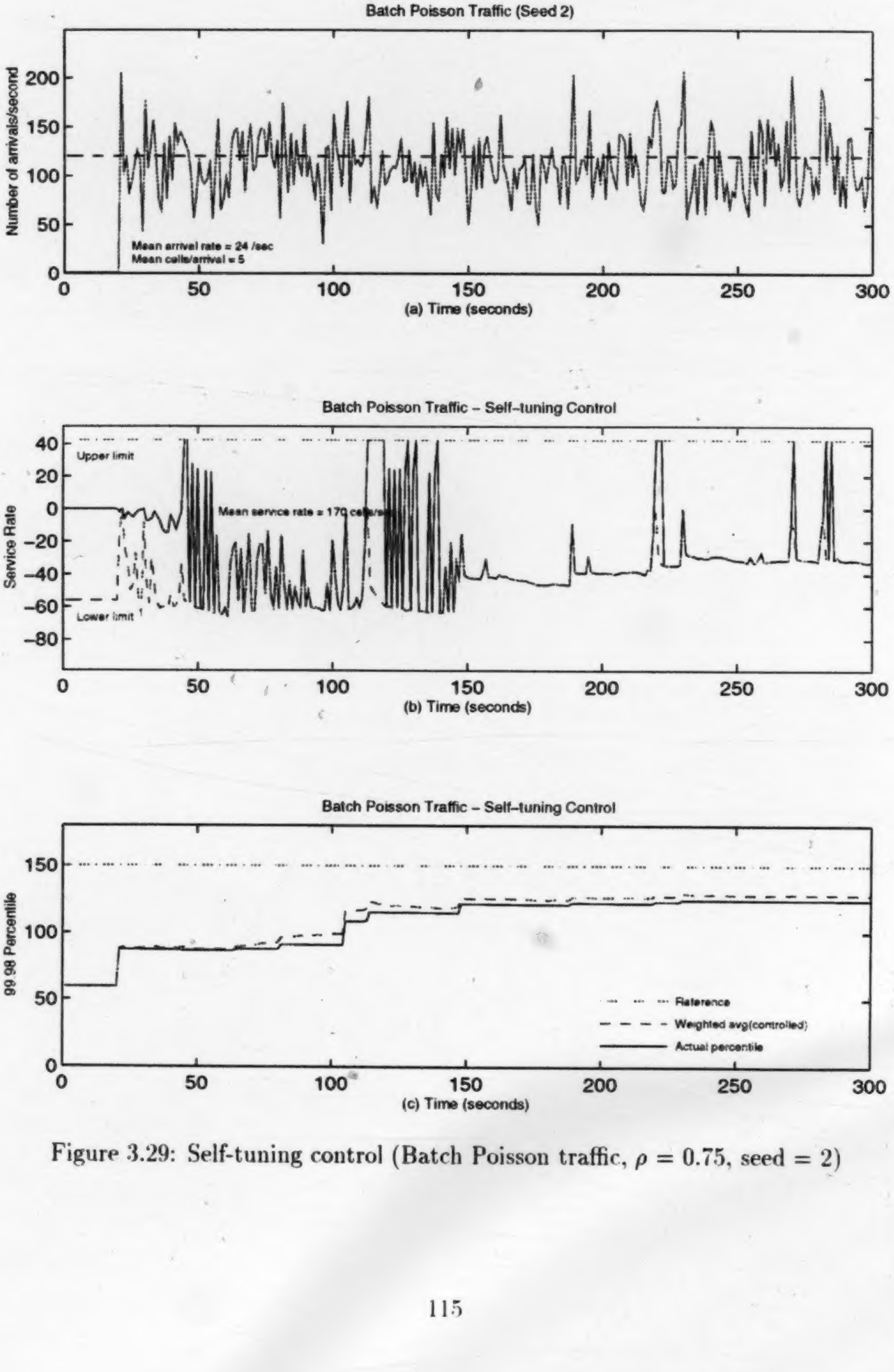


Figure 3.29: Self-tuning control (Batch Poisson traffic, $\rho = 0.75$, seed = 2)

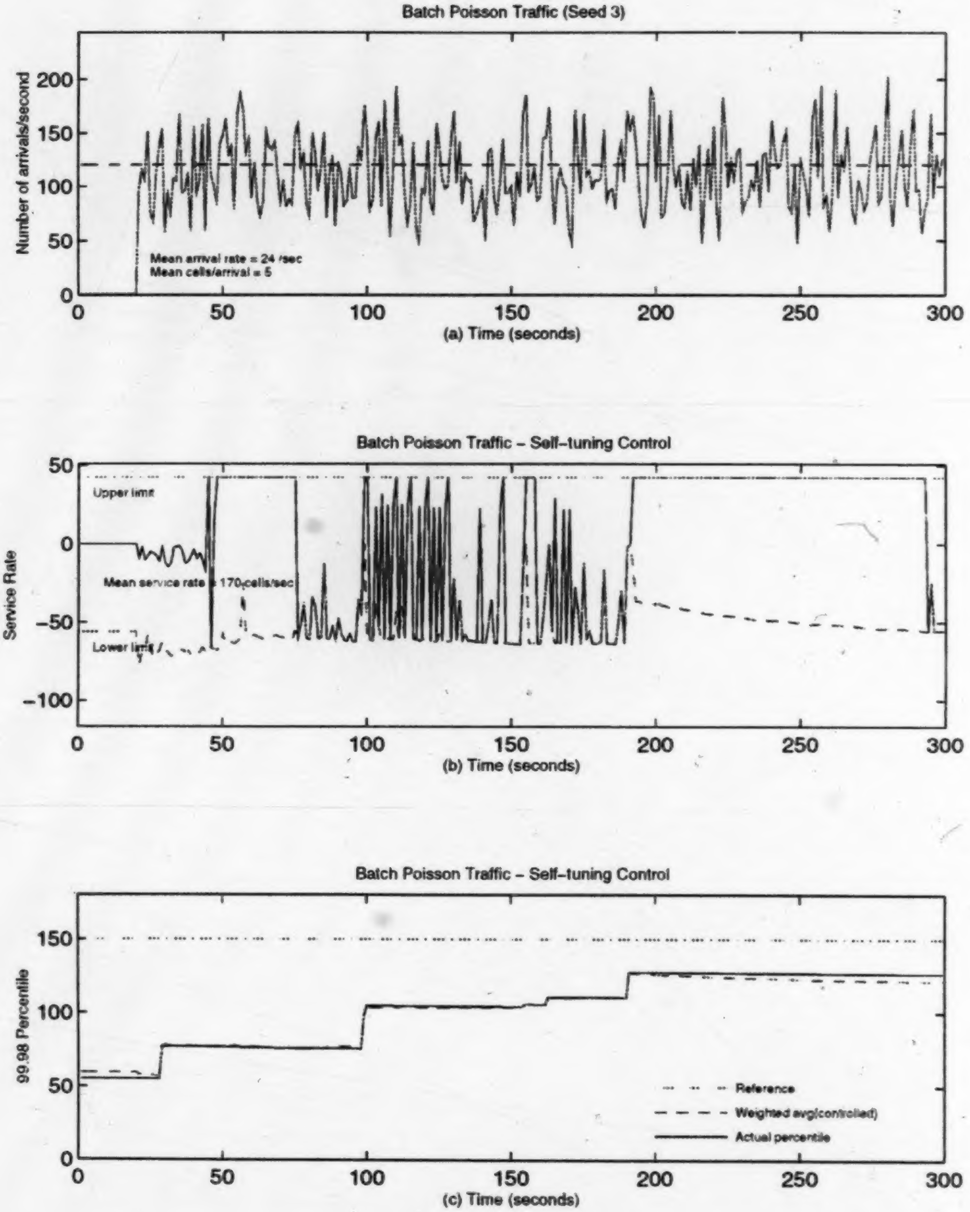


Figure 3.30: Self-tuning control (Batch Poisson traffic, $\rho = 0.75$, seed = 3)

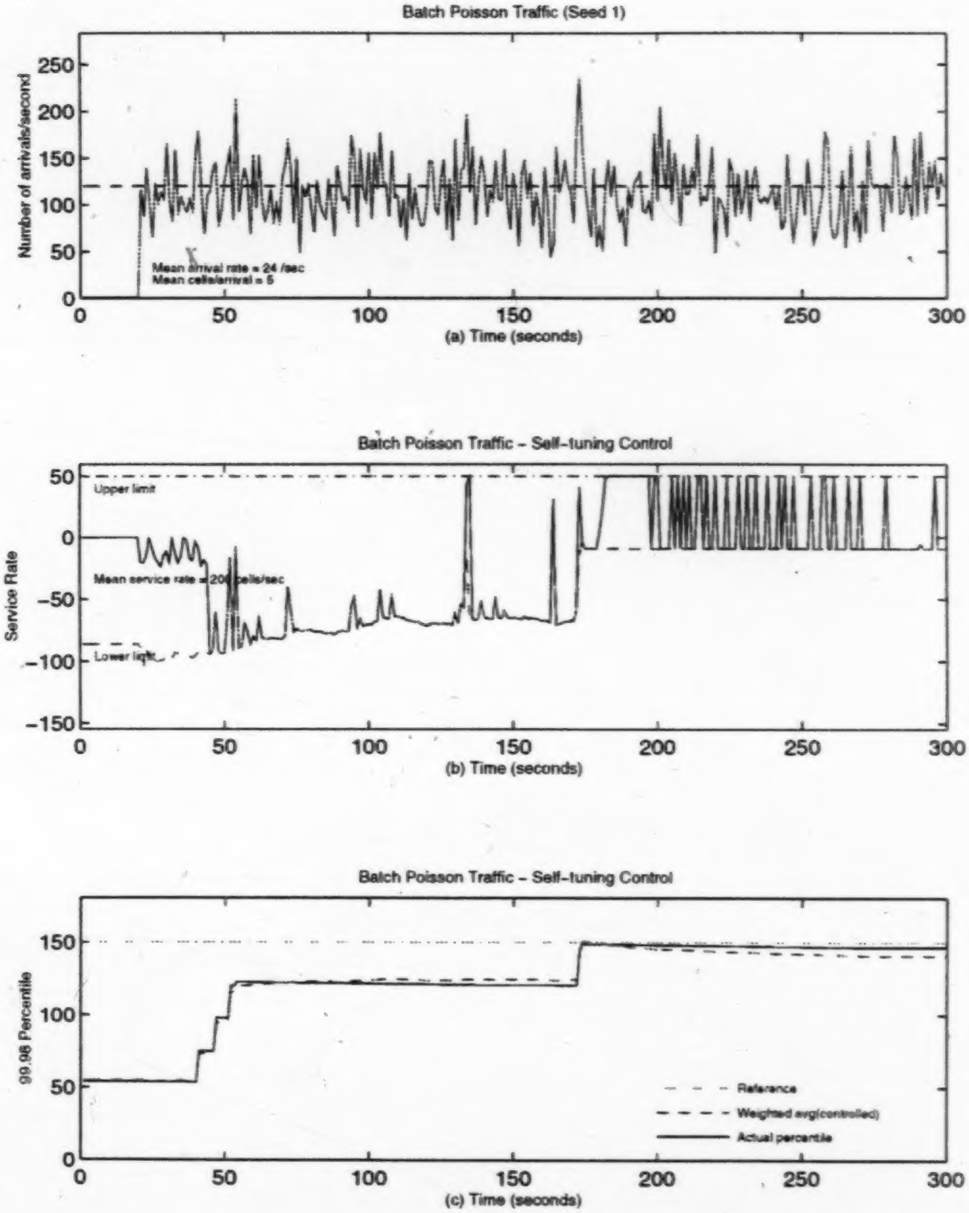


Figure 3.31: Self-tuning control (Batch Poisson traffic, $\rho = 0.6$, seed = 1)

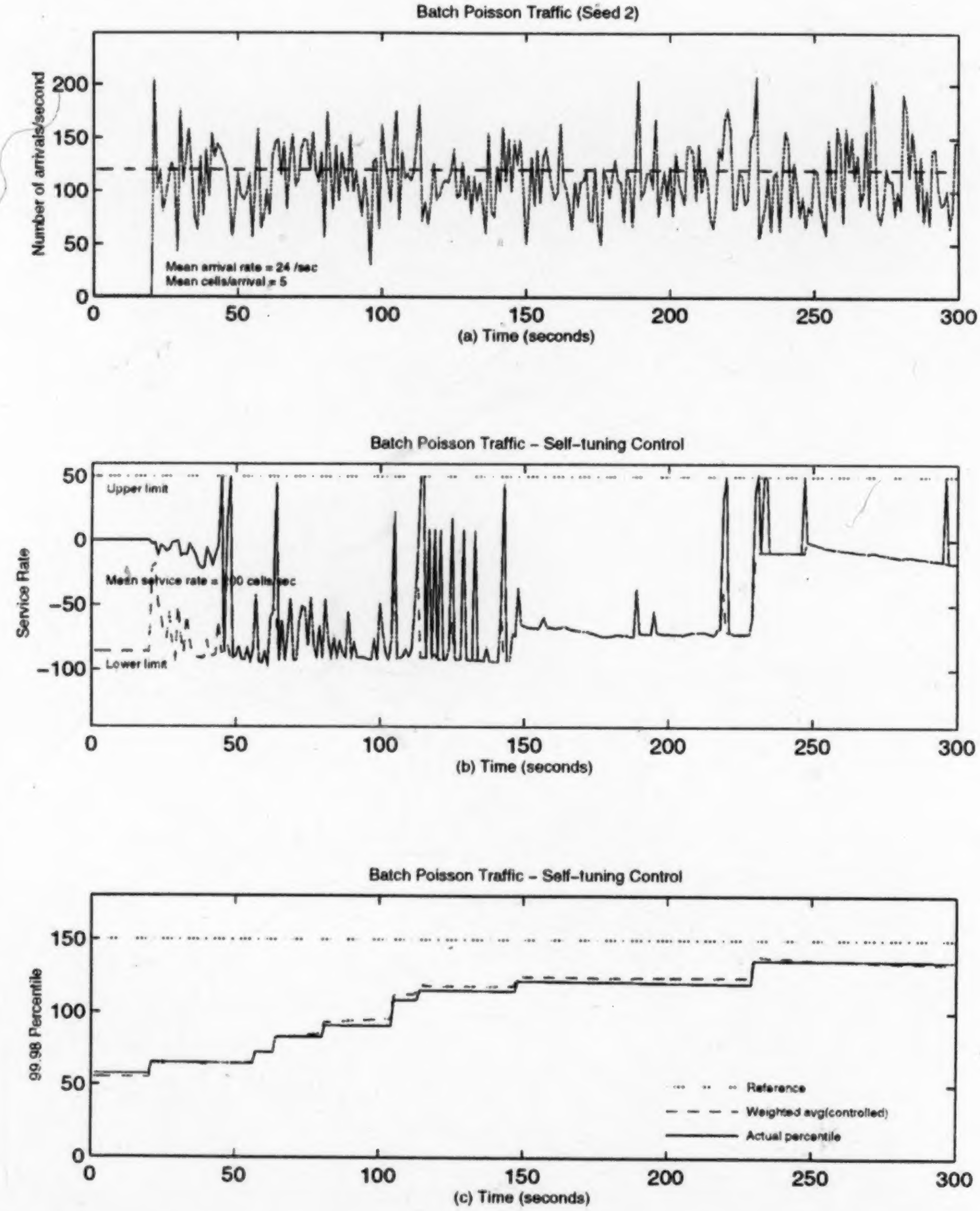


Figure 3.32: Self-tuning control (Batch Poisson traffic, $\rho = 0.6$, seed = 2)

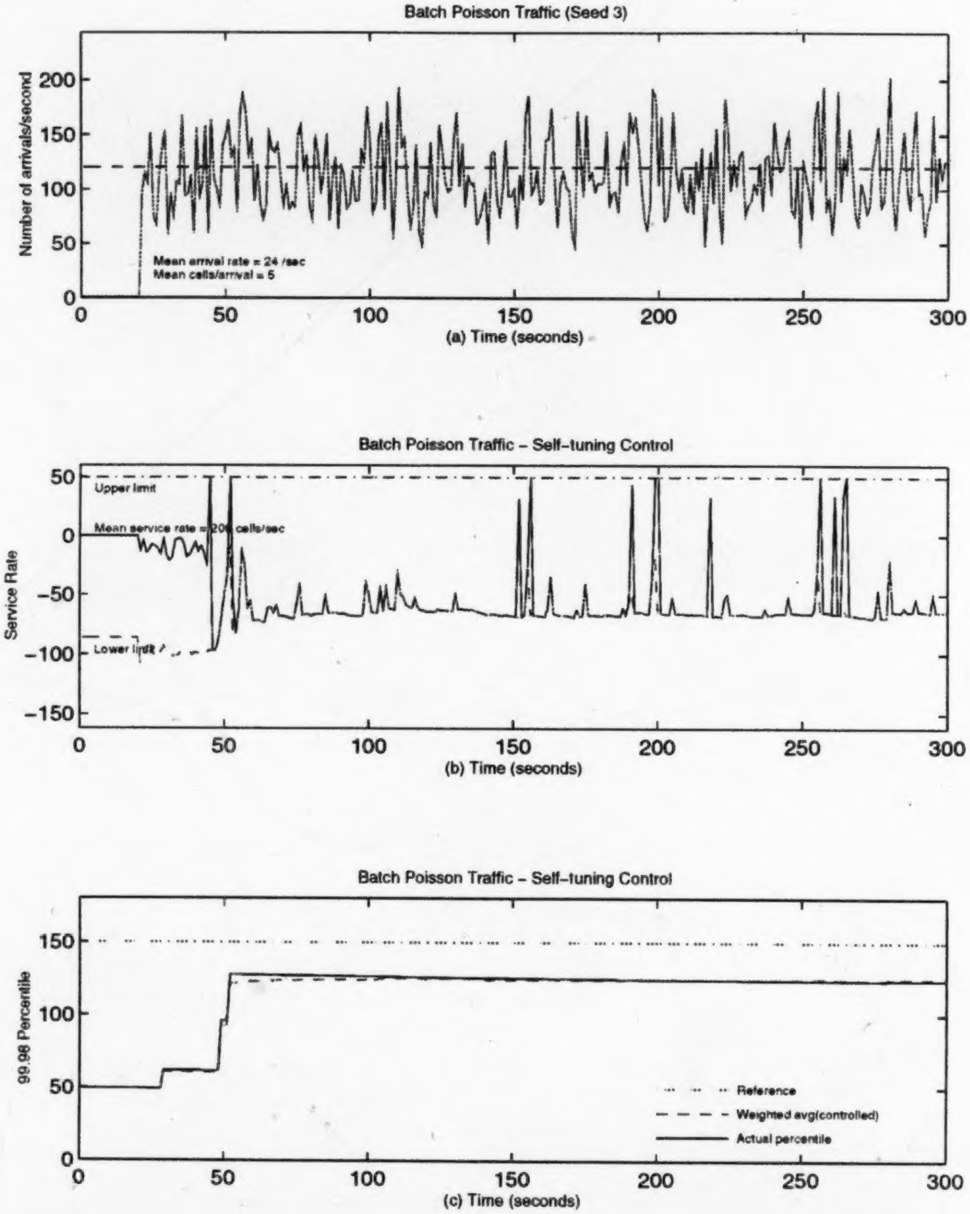


Figure 3.33: Self-tuning control (Batch Poisson traffic, $\rho = 0.6$, seed = 3)

3.9.1.3 Self-similar traffic

In order to investigate the effect of long-range dependence (LRD) on the controller performance, the queue is fed with self-similar traffic. The reference level is chosen to be 250 cells. The initial startup time was increased to 200 seconds as the queue took a longer time to stabilize. The self-similar traffic model has the following parameters:

- The mean arrival rate is 120 cells/sec as usual.
- The variance of the arrival rate is $2\,\mu$. Therefore the maximum variance is 240.
- The Hurst parameter (section 3.6.3) is fixed at 0 7 to generate fairly self-similar traffic.
- The traffic model output is sampled every T_{sample} seconds. The duration of the simulation is chosen to be at least three orders of magnitude higher to make sure of observing LRD effects.

Figures 3.34(c), 3.35(c) and 3.36(c) show the controller performance for $\rho = 0.8$. The error can be seen to be negligible. From sub-figure(b) in each case, we can see that the variability of the traffic is greater than in the Poisson or batch Poisson cases.

Figures 3.37(c), 3.38(c) and 3.39(c) show the controller performance for $\rho = 0.7$. The output can be seen to be very close to the reference value.

Figures 3.40(c), 3.41(c) and 3.42(c) show the controller performance for $\rho = 0.6$. Once again, the error is very small. In figure 3.41(c), we can seen that there is a small overshoot, to which the controller responds correctly by increasing the service rate (sub-figure (b)).

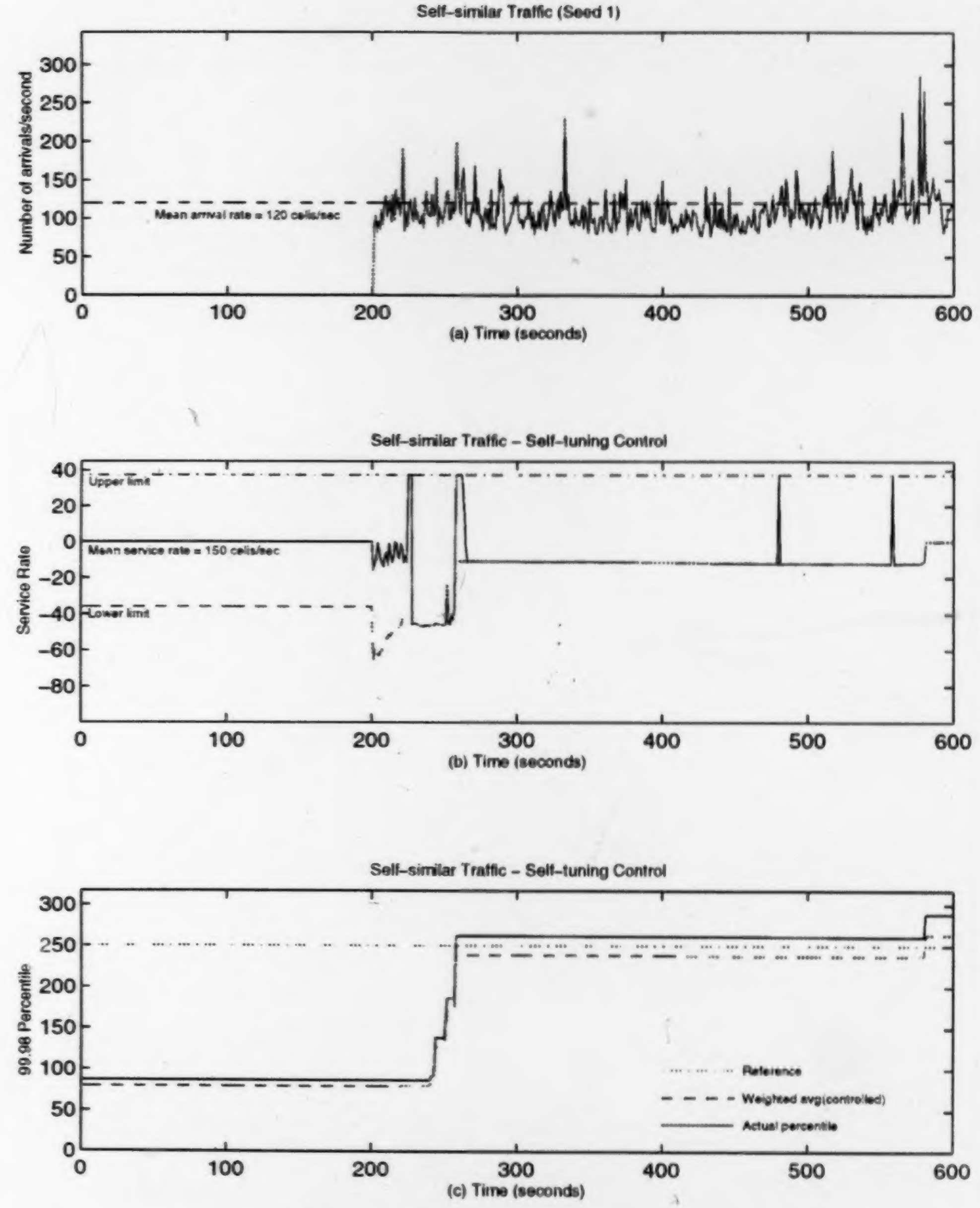


Figure 3.34: Self-tuning control (self-similar traffic, $\rho = 0.8$, seed = 1)

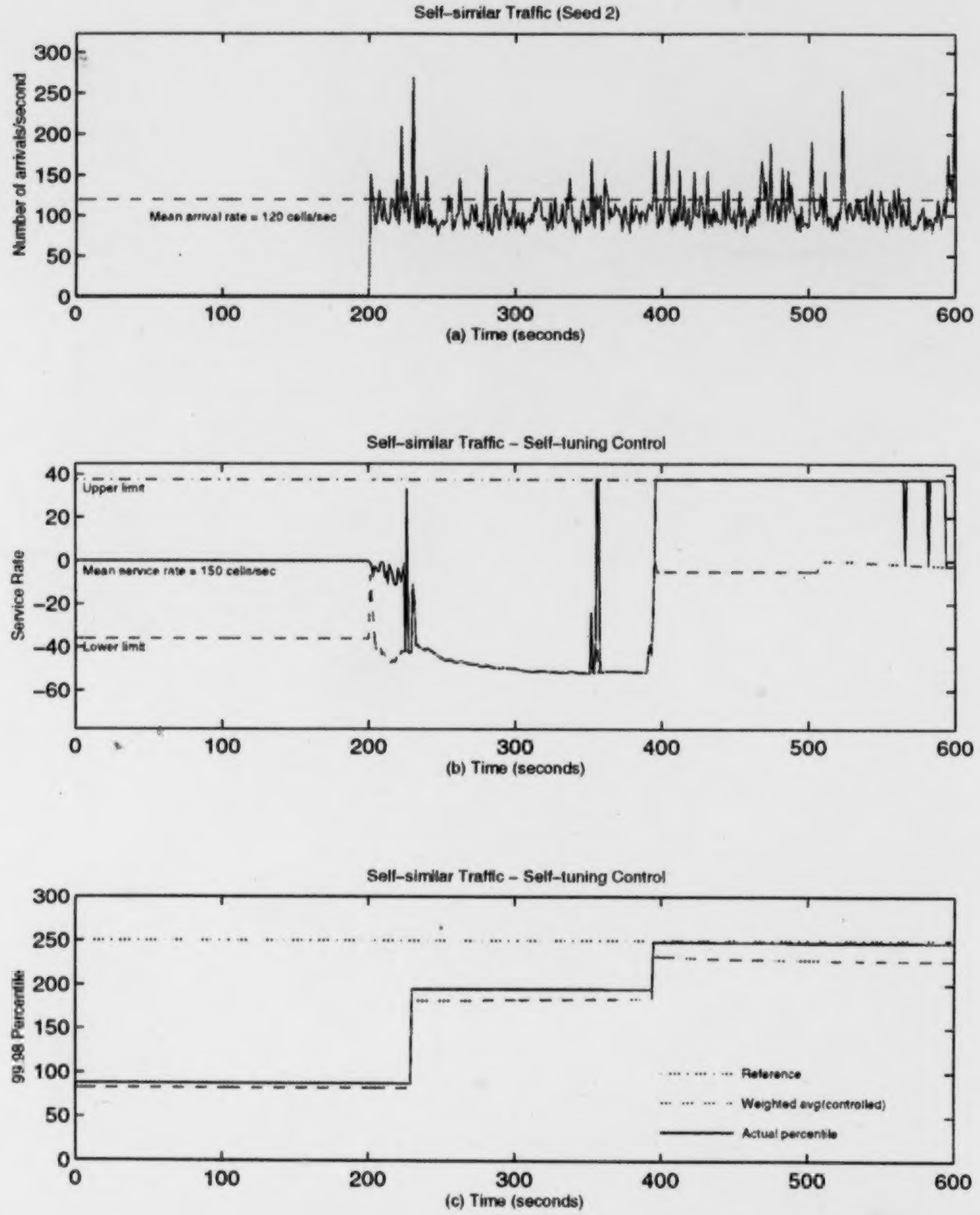


Figure 3.35: Self-tuning control (self-similar traffic, $\rho = 0.8$, seed = 2)

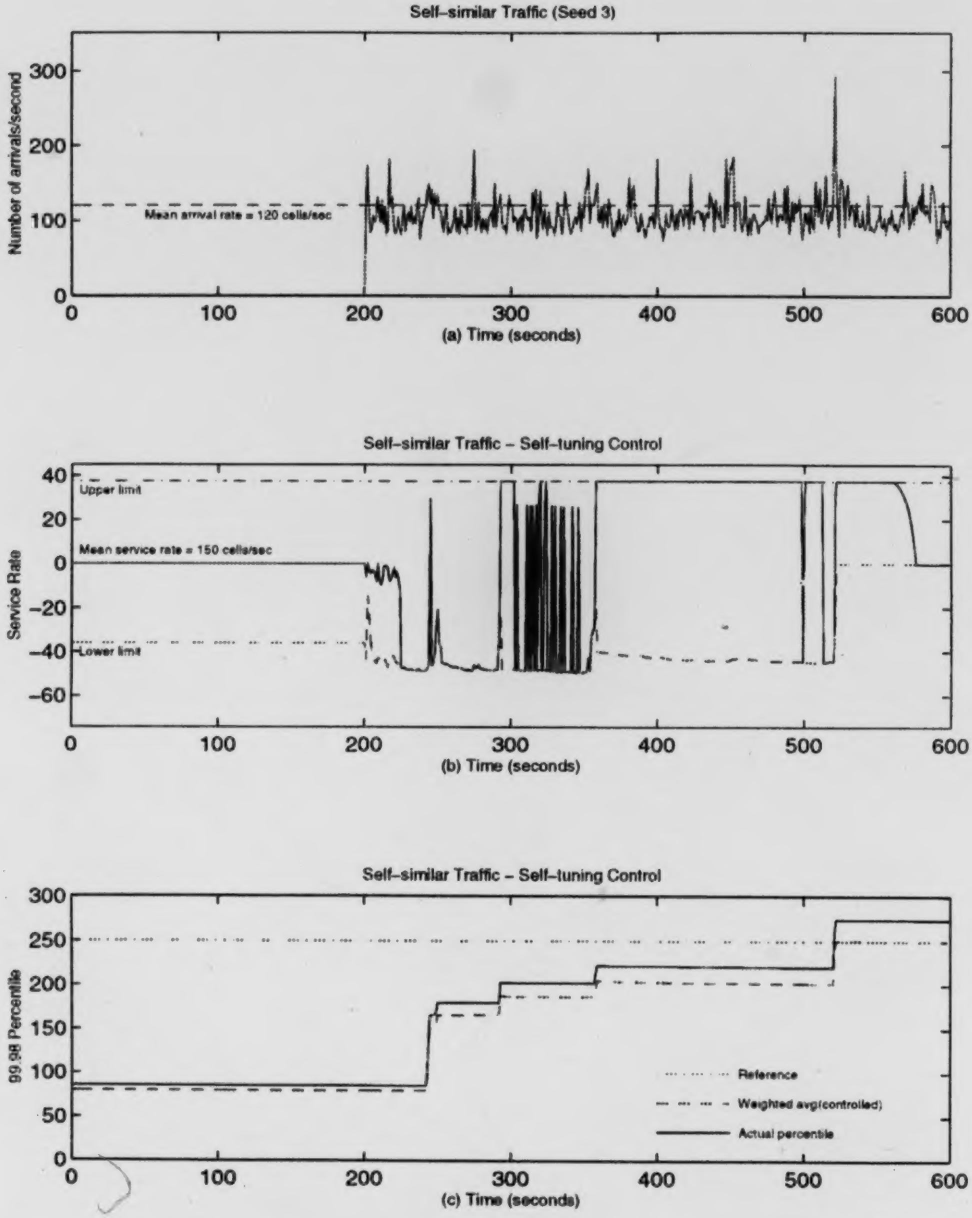


Figure 3.36: Self-tuning control (self-similar traffic, $\rho = 0.8$, seed = 3)

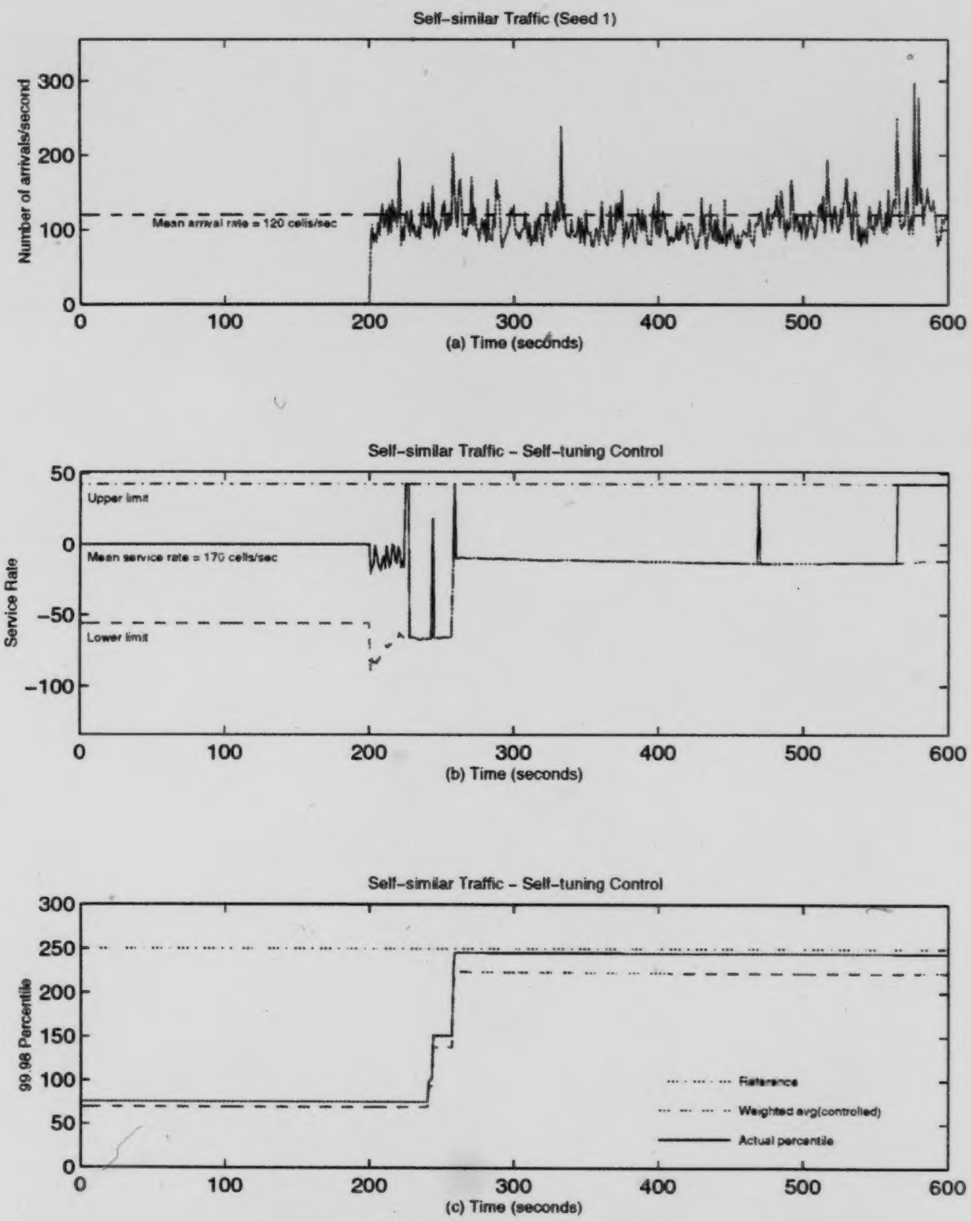


Figure 3.37: Self-tuning control (self-similar traffic, $\rho = 0.75$, seed = 1)

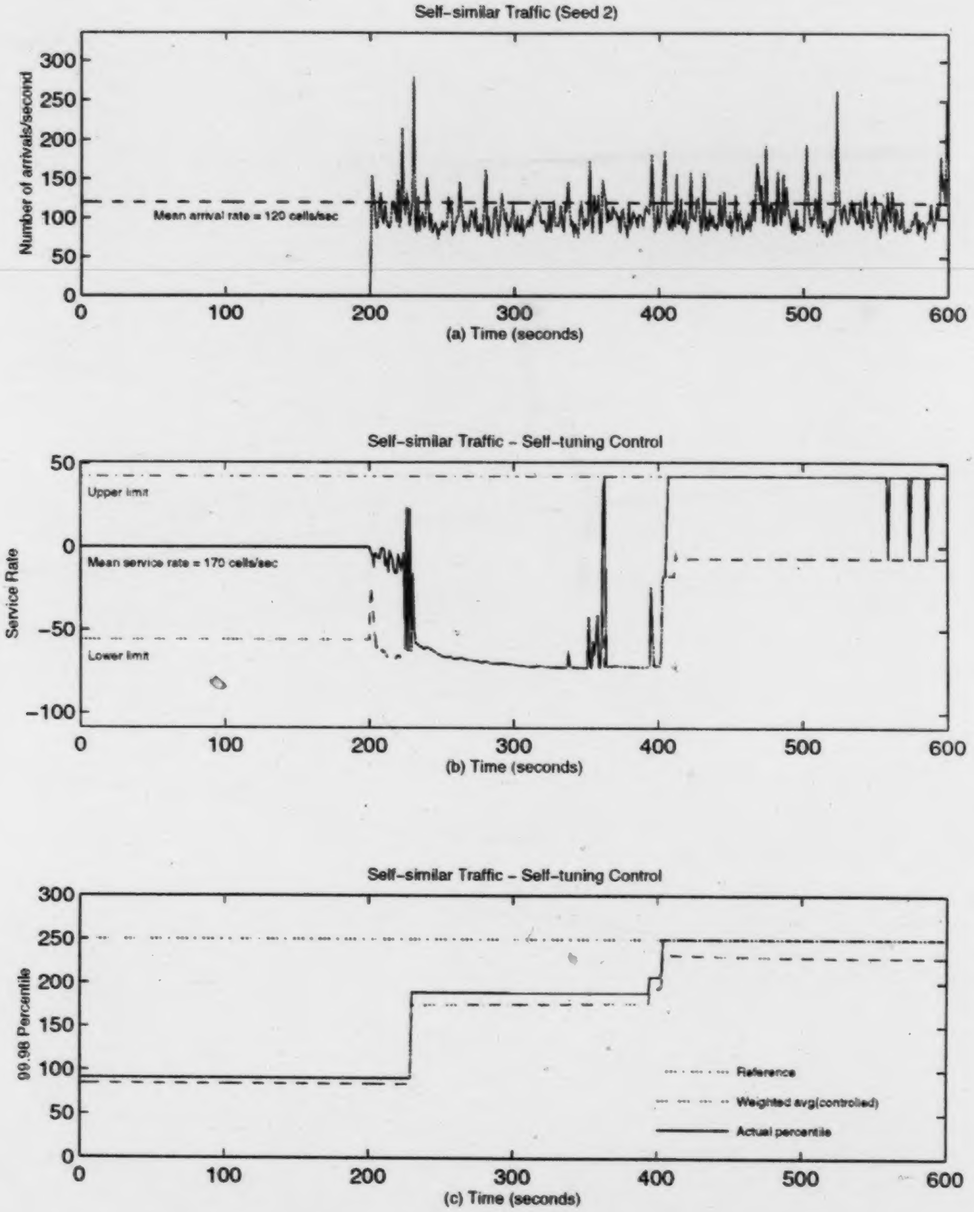


Figure 3.38: Self-tuning control (self-similar traffic, $\rho = 0.75$, seed = 2)

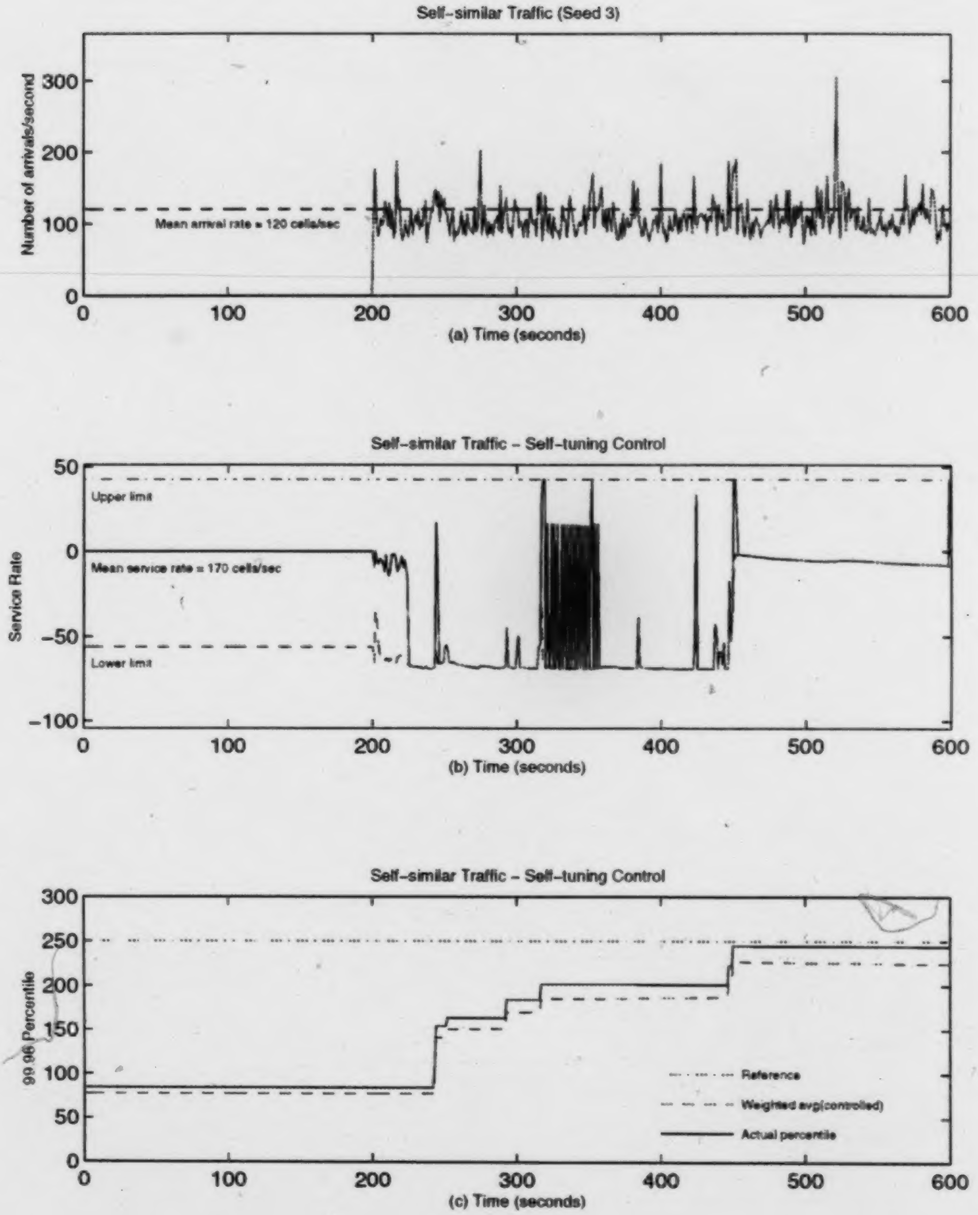


Figure 3.39: Self-tuning control (self-similar traffic, $\rho = 0.75$, seed = 3)

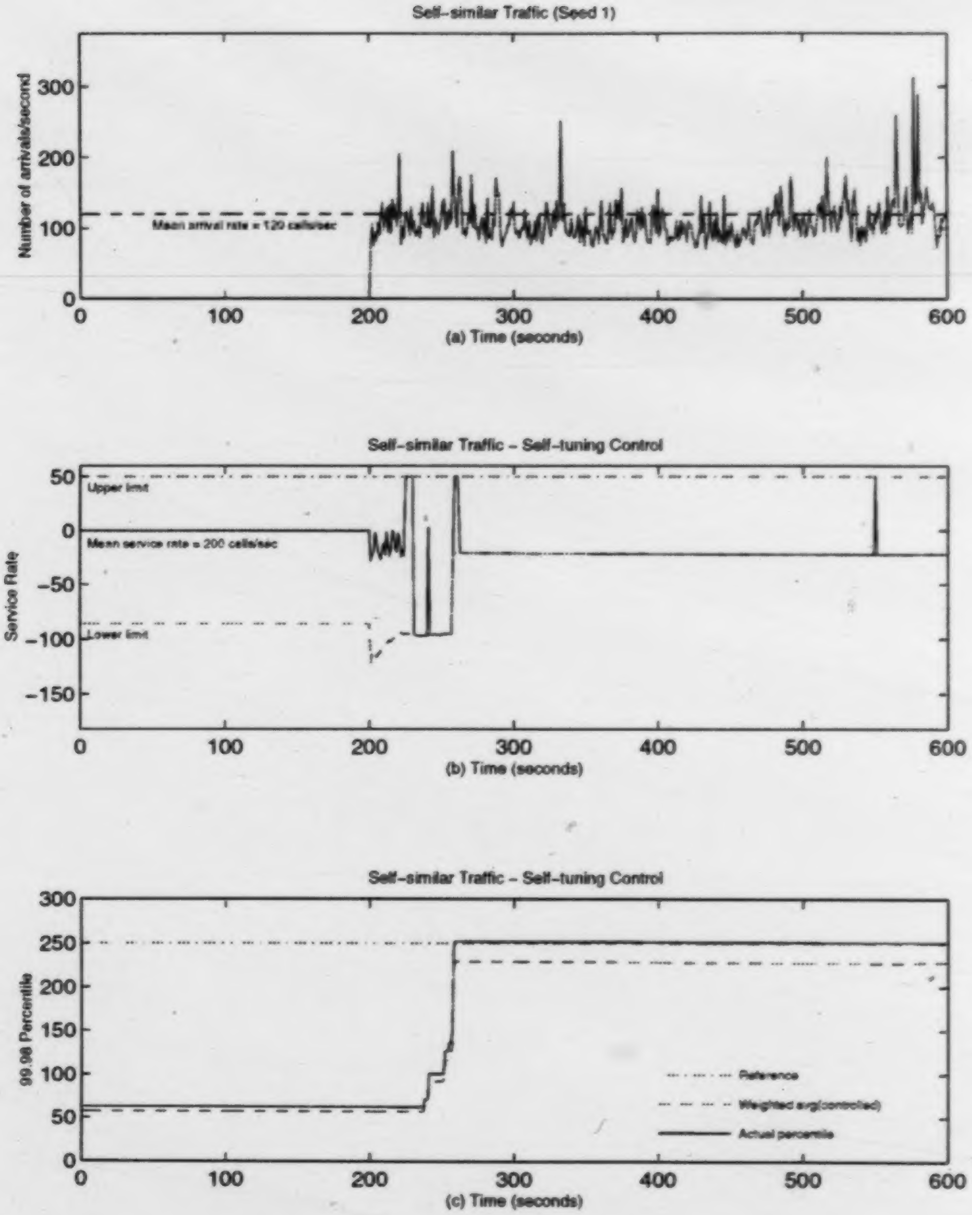


Figure 3.40: Self-tuning control (self-similar traffic, $\rho = 0.6$, seed = 1)

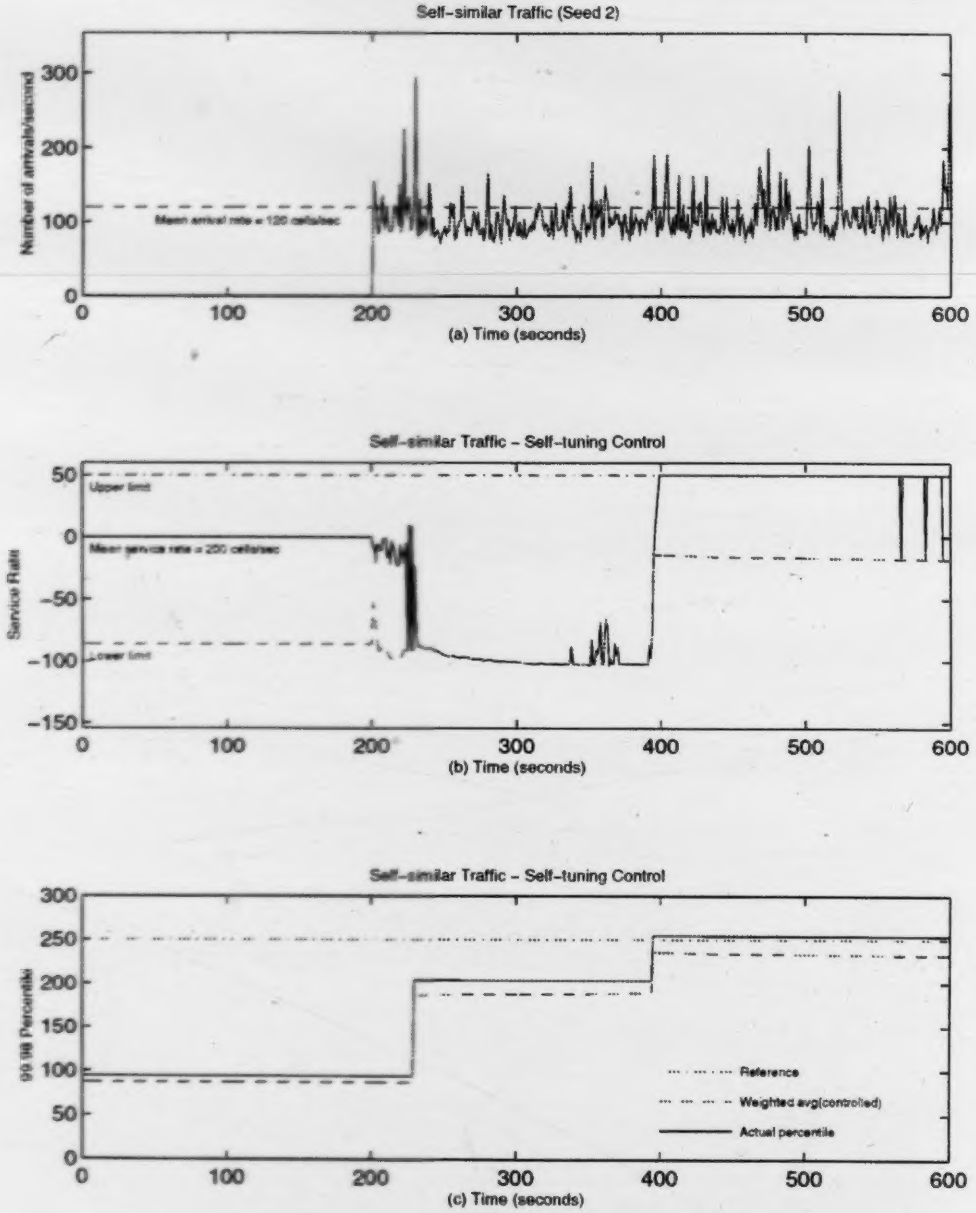
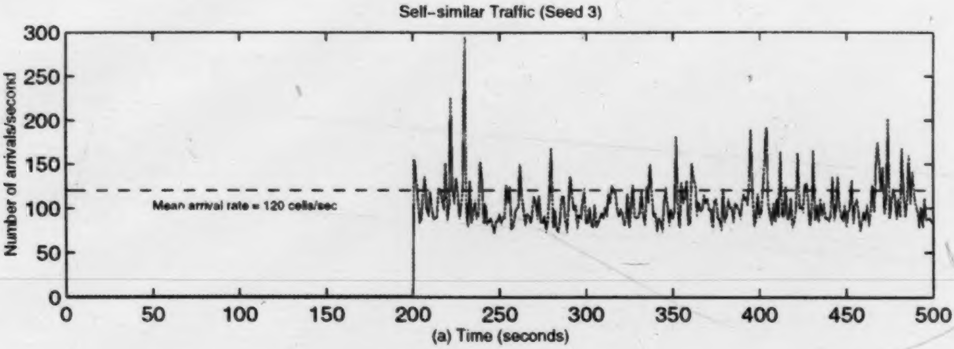
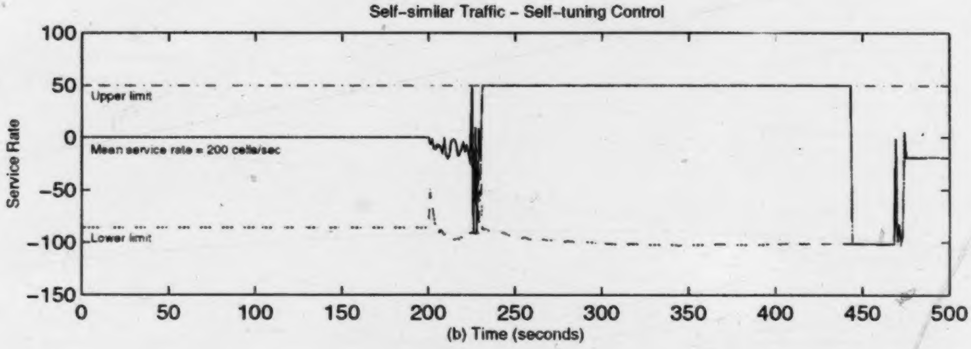


Figure 3.41: Self-tuning control (self-similar traffic, $\rho = 0.6$, seed = 2)





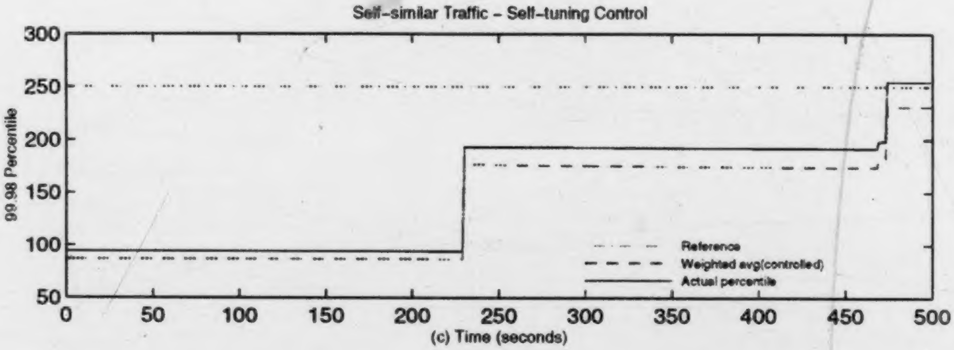


Figure 3.42: Self-tuning control (self-similar traffic, $\rho = 0.6$, seed = 3)

3.9.2 Adaptive controller

One of the advantages of an adaptive controller, as opposed to a controller with a static plant model, is that it is able to maintain control in spite of varying plant parameters. To test this ability, we changed the mean arrival rate during the simulation. In the first test, the arrival rate was changed from 120 cells/sec to 130 cells/sec, causing the utilization to change from 0.8 to 0.87. This change in utilization causes a variation in the plant gain. The corresponding change in the buffer quantile when the plant is in open loop mode (no controller) is shown in figure 3.43. It can be seen that the change in arrival rate causes a small change in the plant output when in open loop,

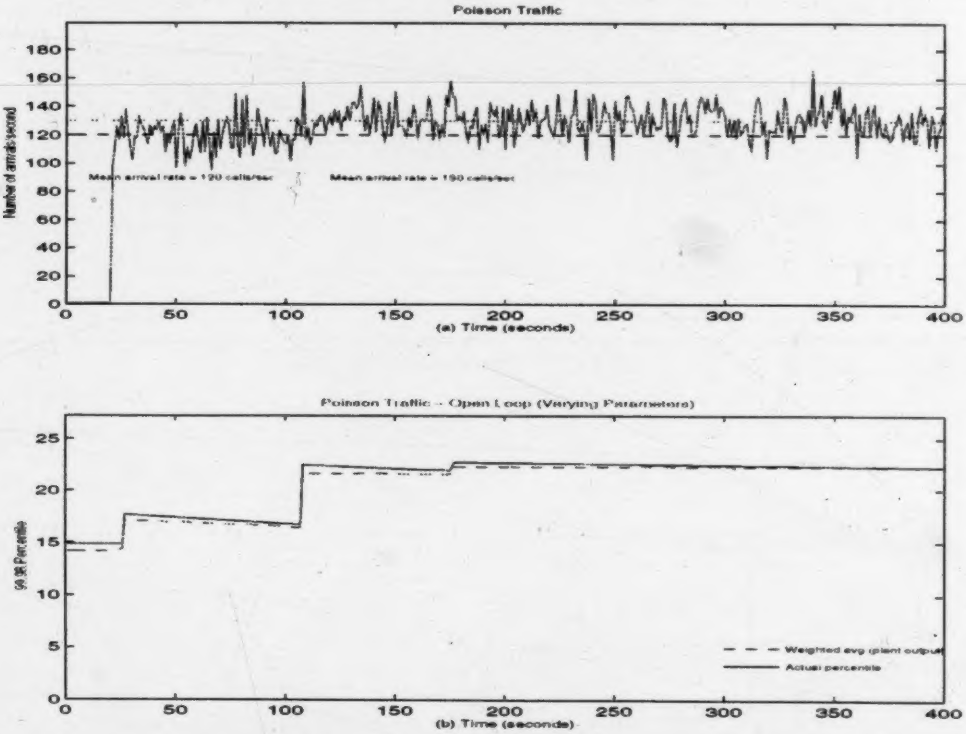


Figure 3.43: Open loop test (varying parameters)

The online identification process under closed loop control should be able to iden-

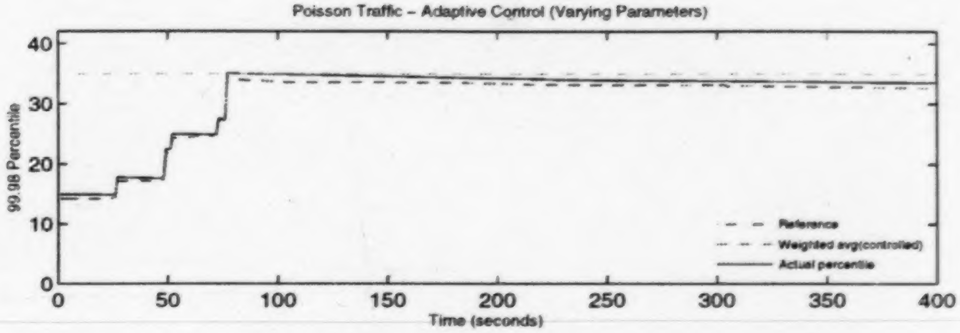


Figure 3.44: Adaptive control (effect of varying parameters)

tify the new gain parameter. The results of the experiment are shown in figure 3.44. The effect of the variation is not visible during closed loop control. Thus, the controller is able to maintain control in spite of varying parameters as well as disturbances.

In the second experiment, the arrival rate is changed from 120 cell/sec to 140 cells/sec. The open loop results show that the buffer quantile jumps to 55 cells as a result of the increased queue utilization (figure 3.45). However, the adaptive controller restricts the overshoot to within 15% of the reference level (figure 3.46).

3.9.3 Effect of forgetting factor

As mentioned in section 3.3.2.3.2, the purpose of the forgetting factor is to discount older information and give greater weight to recent information. This allows the controller to adapt quickly to changes in the plant behavior. The forgetting factor $\lambda \in (0, 1]$, and is usually selected from the interval $[0.95, 1]$.

Values of λ close to 1 enable the identification routine to "remember" to a greater extent information from the past while smaller values enable it to "forget" more quickly. However, smaller forgetting factors also make the controller "jumpy". Because a smaller amount of information is used to build the model, the identified model changes more quickly. Hence, the control effort is likely to fluctuate more.

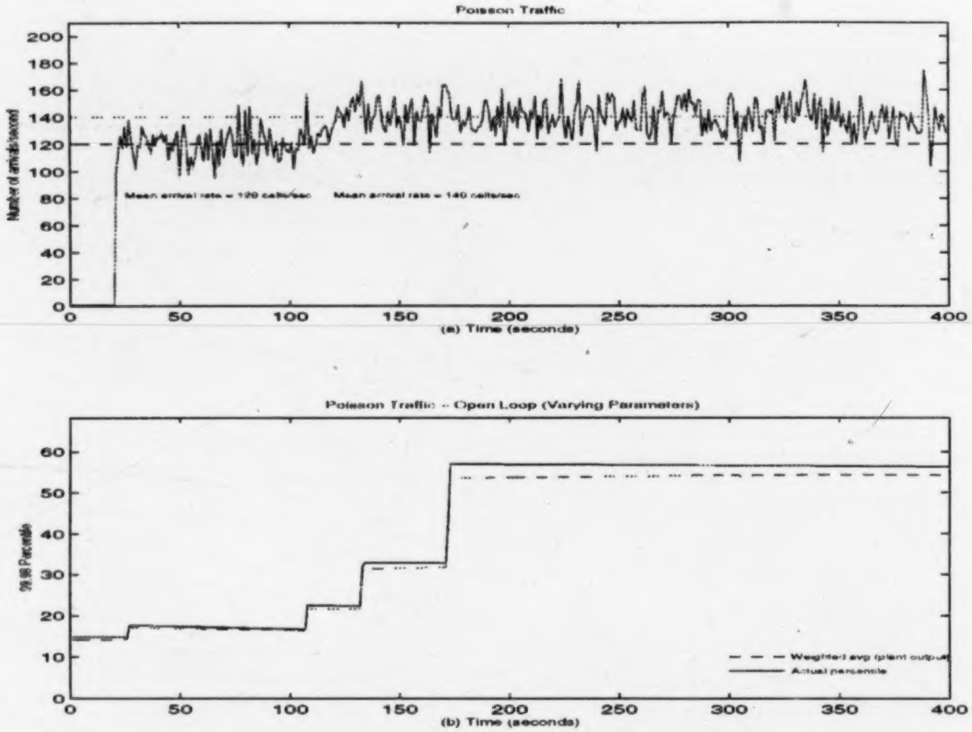


Figure 3.45: Open loop test (varying parameters)

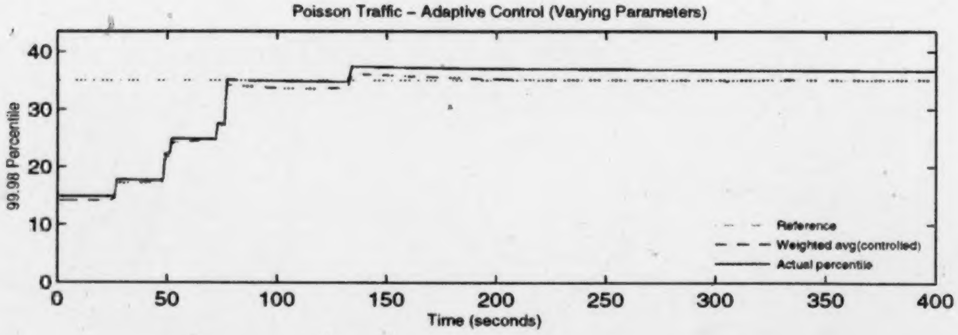


Figure 3.46: Adaptive control (effect of varying parameters)

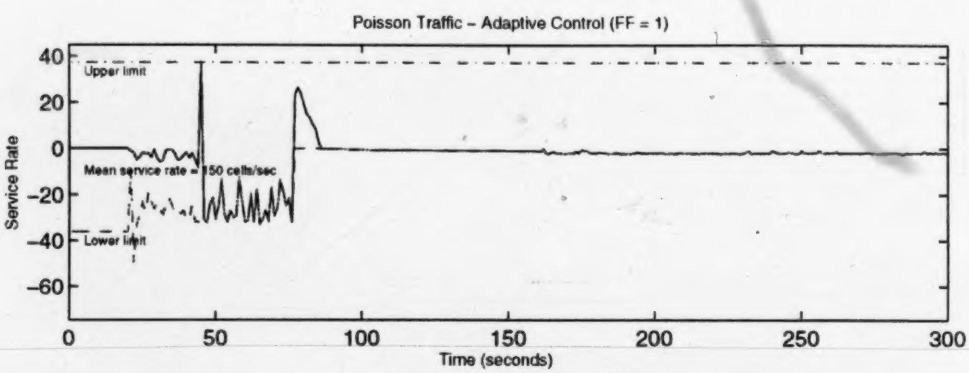


Figure 3.47: Self-tuning control (effect of forgetting factor, $\rho = 0.8$, FF = 1)

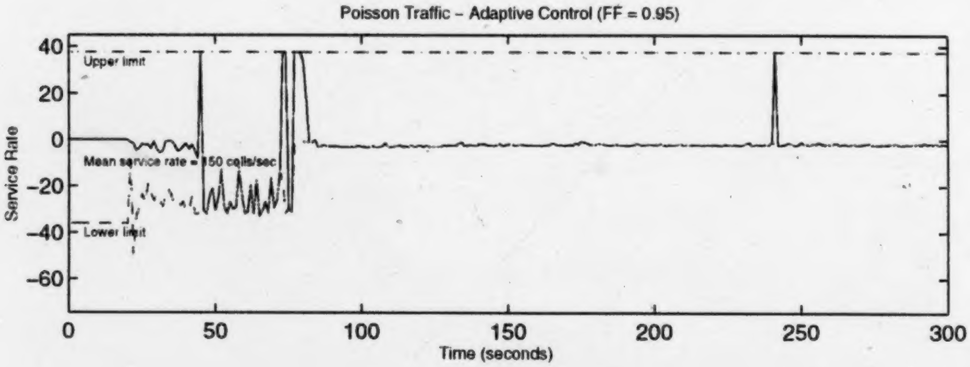


Figure 3.48: Self-tuning control (effect of forgetting factor, $\rho = 0.8$, FF = 0.95)

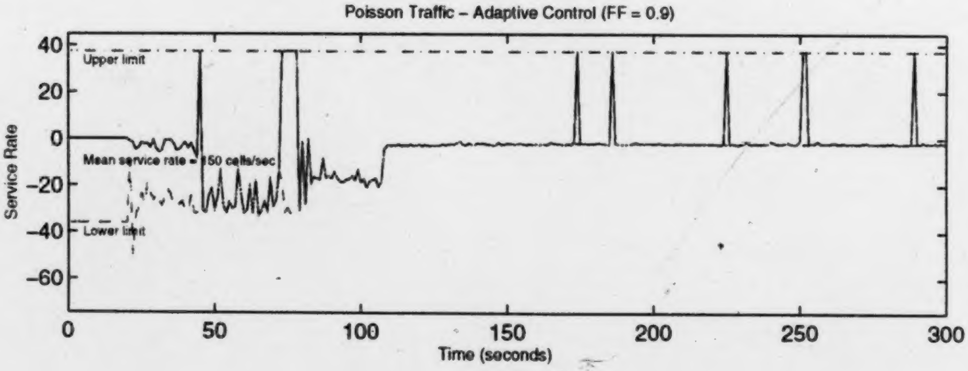


Figure 3.49: Self-tuning control (effect of forgetting factor, $\rho = 0.8$, FF = 0.9)

Figures 3.47, 3.48 and 3.49 show the effect of reducing the forgetting factor for $\lambda = 1, 0.95$ and 0.9. It can be seen that the control effort fluctuates more for $\lambda = 0.95$ than for $\lambda = 1$ (time > 150 seconds). Similarly, the control effort variation is greater for $\lambda = 0.9$ than for $\lambda = 0.95$. The mean service rate in these experiments was 150 cells/sec.

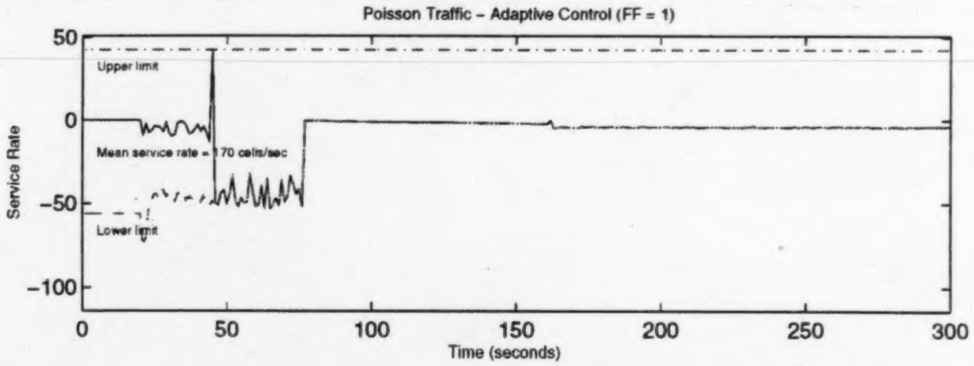


Figure 3.50: Self-tuning control (effect of forgetting factor, $\rho = 0.75$, FF = 1)

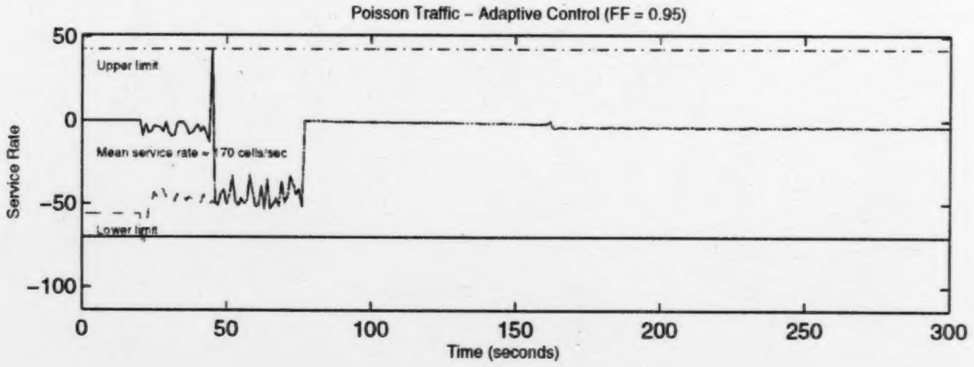


Figure 3.51: Self-tuning control (effect of forgetting factor, $\rho = 0.75$, FF = 0.95)

In the next set of experiments, the mean service rate is increased to 170 cells/sec. It can be seen from figures 3.50, 3.51 and 3.52 that while the control effort does not vary significantly when the forgetting factor changes from 1 to 0.95, there is greater fluctuation when $\lambda = 0.9$.

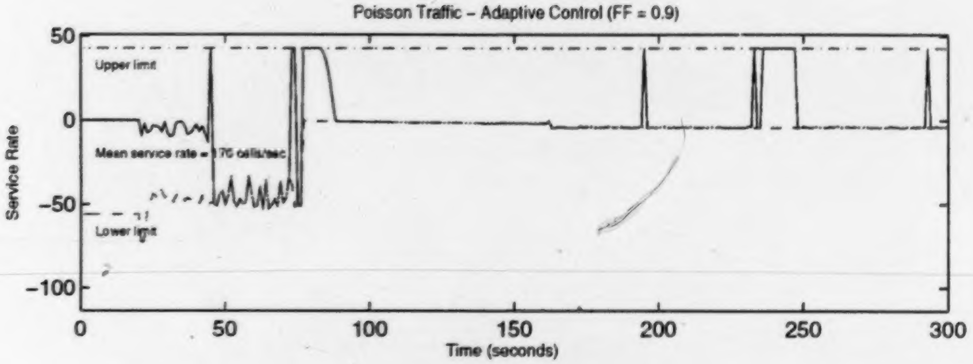


Figure 3.52: Self-tuning control (effect of forgetting factor, $\rho = 0.75$, FF = 0.9)

3.9.4 Effect of mean service rate

In this section, we investigate the effect of the CAC service rate allocation, μ, which is used as the mean value for service rate modulation by the controller. Looking at the results of the previous experiments, we can see that initially the controller lowers the service rate to increase the buffer quantile to the reference value. However, when the buffer quantile approaches the reference value, the lower constraint on the service rate, μ_t^{min} rises closer to the mean value. As a result, when the quantile is close to the reference value, the instantaneous service rate, μ_t is close to the mean value μ. Since the quantile decreases very slowly, for a very long time it will be the case that $\mu_t \sim \mu$.

Since the average delay for the controller depends on the mean service rate, we can conclude that if the buffer quantile is close to the reference value, the average delay is a function of the service rate allocated by CAC. By controlling both the mean and a specified quantile of the buffer occupancy distribution, the shape of the distribution can be shaped to a desired specification, giving greater control over the QoS than could be achieved by controlling the mean or a specified quantile of the

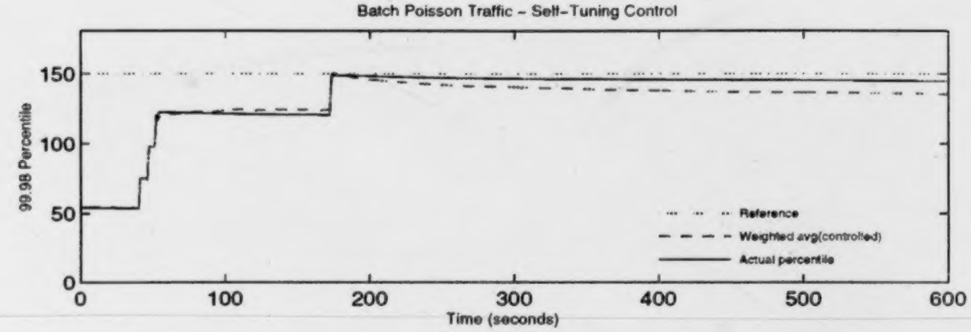


Figure 3.53: Self-tuning control (effect of mean service rate, $\rho = 0.6$)

buffer occupancy in isolation.

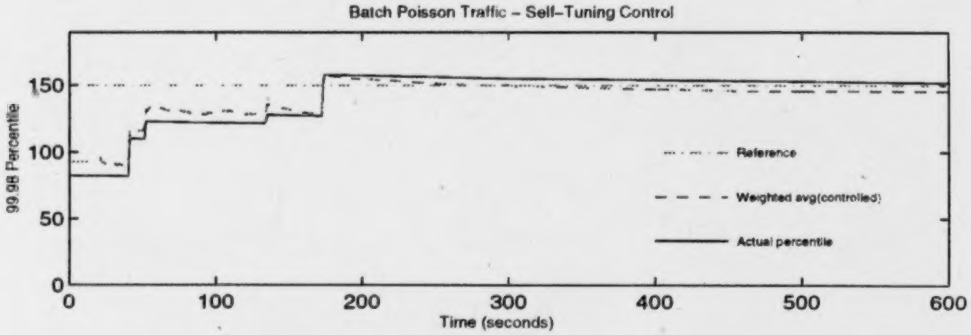


Figure 3.54: Self-tuning control (effect of mean service rate, $\rho = 0.8$)

To verify this, we conducted two experiments. The queue was excited with batch Poisson traffic with a mean arrival rate in both cases of 120 cells/sec. The mean service rate in the first case was 150 cells/sec and 200 cells/sec in the second. Figures 3.53 and 3.54 show that the 99.98 quantile of the buffer occupancy is close to the reference value in both cases.

The cumulative distribution function (CDF) is plotted against the buffer index in each case (figure 3.55). At lower buffer positions, the plot for the service rate of

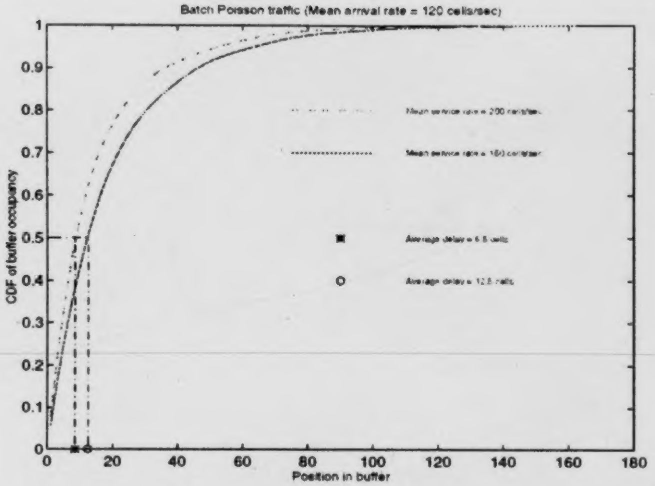


Figure 3.55: Effect of mean service rate on buffer occupancy CDF

200 cells/sec is shifted to the left of the plot for 150 cells/sec. In particular, the average delay for $\mu = 150$ is 12.5 cells while that for $\mu = 200$ is only 8.5 cells, a 32% improvement achieved by the use of a higher service rate. This demonstrates that we can independently match the average delay and quantile to specified values by choosing an appropriate mean service rate and then applying the feedback control.

3.10 Conclusions

We have described a cell scheduler for a queue fed with aggregate traffic from a group of multiplexed VBR calls. The scheduler uses adaptive feedback control to maintain the QoS at a desired level, in spite of traffic fluctuations and/or variations in the aggregate traffic characteristics.

The scheduler modulates the service rate in order to control the QoS. This local control is superior to flow control of the individual sources generating traffic into the queue—in addition, it allows the scheme to be used in WANs with large propagation delays.

We have demonstrated that the scheme is robust in its ability to control widely varying traffic types—without any modification whatsoever to the controller—using Poisson, batch Poisson and self-similar traffic models. This is a significant improvement over the scheme in [29], where the control effort weighting parameter (denoted there by λ) must be re-tuned whenever the traffic characteristics change. The improvement is obtained by the use of a minimum-bias control ([19]).

We have also shown that the controller is able to maintain the QoS at the desired level inspire of bit rate variations. Good control is achieved even at high utilization of 80%. The controller is also able to adapt when the mean arrival rate deviates from its intended value.

In [29], it is shown that the long-term QoS converges to the desired level. However, if the call duration is not long enough, the QoS provided may be very different from the requested value. The use of the weighted average of the measured buffer quantile and the calculated buffer quantile prevents the QoS for calls of short duration from being violated

The calculated quantile is obtained using the mean and variance of the net arrival process—the auto-covariance sum is not calculated—simplifying the amount of computation that must be performed at each sampling instant. The controller performance was found to be adequate enough that enhancements in the form of feed-forward control were not carried out. This results in a simpler controller compared to that described in [29].

The use of control methodology makes it possible to utilize the plethora of online process monitoring techniques. The model of the plant estimated at each sampling instant can be examined for stability. Local congestion can therefore be detected and corrected by the network operator. Large deviations of the plant output from the reference value—especially when the QoS exceeds the reference value by large amounts—can be used to generate alarms, requesting manual intervention.

The initial bandwidth allocated by CAC is used as the mean level about which the instantaneous service rate is varied to maintain the QoS. The choice of the initial bandwidth allows the CAC to control the average delay, while the maximum delay, delay jitter and cell loss ratio are controlled by the scheduler. The performance of the controller was found to be fairly insensitive to the choice of forgetting factor.

Chapter 4

Dynamic call admission control

4.1 Introduction

B-ISDN networks are intended to support traffic of several different classes, each with its own QoS requirement. With cell multiplexing, a high network utilization can be achieved, but sophisticated congestion control techniques are needed to ensure that the QoS of the individual traffic sources is never violated. Call admission control (CAC) can be described as a preventive congestion control that restricts the admission of new calls to those that will not cause violation of the QoS of previously admitted calls.

As mentioned in section 2.1, many call admission schemes ([4, 15, 31]) are based on the concept of *effective bandwidth* or equivalent capacity: the effective bandwidth of a traffic source is defined as the bandwidth required to guarantee a specific QoS. Such schemes are *static*: the bandwidth requirements of a group of multiplexed calls is a function of their individual effective bandwidths alone.

This makes CAC schemes based on EB particularly attractive since the admission policy does not need to consider traffic in other queues (belonging to other classes) to decide whether an incoming call should be admitted. However, admission policies that do take traffic in other queues into account have the potential to improve on the static schemes. We call them *dynamic* CAC schemes, since the bandwidth allocated to a particular group of calls is non-deterministic.

The traffic arriving at a switch is assumed to belong to one of two classes: variable bit rate (VBR) traffic—for which the QoS consists of delay, delay jitter and cell loss rate bounds—and available bit rate (ABR) traffic whose QoS is specified in terms of a cell loss rate alone.

We propose a dynamic CAC (D-CAC) that reduces the call blocking rate for VBR traffic (α_{VBR}). The D-CAC is not a stand-alone scheme; it overlays a static CAC (S-CAC) in order to improve on a chosen aspect of the latter's performance. We would like to improve α_{VBR} for the following reason: since VBR traffic has more stringent QoS requirements than ABR traffic, it is likely that the network will charge a higher price for a VBR call, just as a courier service with delivery guarantees is more expensive than regular mail. In such a situation, it is clear that higher profits can be achieved by the network provider, if the link is used to a greater extent by VBR traffic. However, if ABR traffic is allowed to compete on an equal footing with VBR traffic—as is the case in current CAC schemes—ABR traffic may cause VBR calls to be blocked. Therefore, we seek to improve the call blocking rate for VBR traffic by giving it priority at the call admission level.

D-CAC has the added advantage that the delay and delay jitter experienced by VBR traffic will be lower than that due to the underlying static scheme (S-CAC)—at low to medium network loads. This is because of over-allocation of bandwidth to the VBR queue. At high loads, this extra bandwidth is traded to admit additional calls—up to a limit indicated by S-CAC.

The rest of the chapter is organized as follows : section 4.2 discusses the system under study. Section 4.3 presents the tradeoff plot—a graphical illustration of the possibility to trade resources (bandwidth, buffers) while maintaining the QoS at a desired level. D-CAC and its properties are described in section 4.4. Simulation studies to compare D-CAC with the S-CAC (on which the D-CAC is based) were performed. The results are presented in section 4.5.1. Section 4.5.4 studies the performance of a modified D-CAC scheme designed to bound the impact of D-CAC on the call blocking

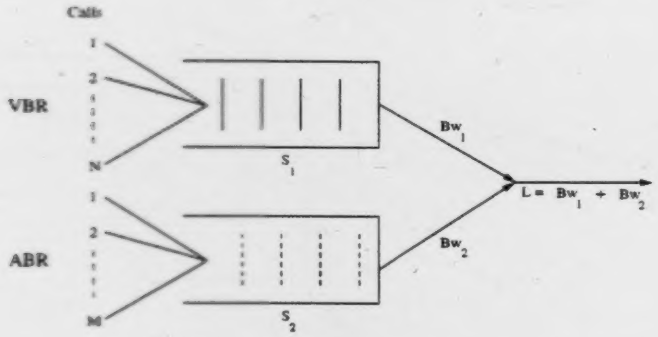


Figure 4.1: System model

rate for ABR traffic.

4.2 System model

Figure 4.1 shows the system to be studied which consists of an output port at an ATM switch. As mentioned in section 1.2, the traffic leaving the port belongs to either the VBR or ABR service class. Incoming VBR traffic is multiplexed onto a single queue while ABR traffic is multiplexed in another. A weighted fair queuing (WFQ) [10] or similar mechanism is used to guarantee each queue the bandwidth allocated to it by CAC. Bandwidth unused by VBR is utilized by ABR traffic.

In this study, calls being multiplexed in the same queue are assumed to have the same QoS requirement. ABR calls have a minimum bandwidth requirement while the bandwidth requirement of a VBR call—the effective bandwidth—can lie between its peak and average cell rate. Switch buffers are divided among the output ports. The buffers allocated to an output port are statically apportioned among the two queues.

4.3 Bandwidth-buffer tradeoff

Link bandwidth and buffers are the resources at each output port shared by the two traffic classes. Since VBR traffic is delay-sensitive while ABR traffic is sensitive to cell loss, it should be possible to allocate a larger share of the bandwidth than strictly

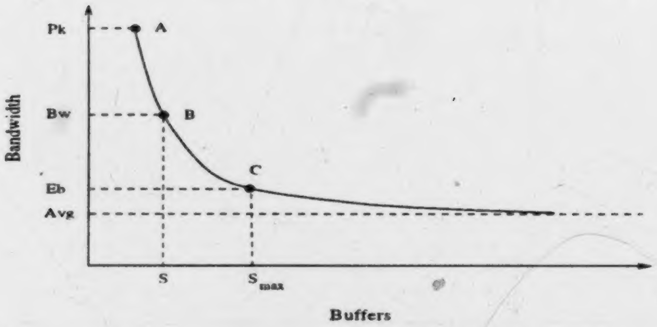


Figure 4.2: Hypothetical resource tradeoff plot for a VBR source

necessary to VBR—giving the queue for ABR traffic a large buffer size—and still meet the QoS needs for both classes. There is therefore more than one possible bandwidth-buffer allocation that will make it possible to meet the desired QoS for the two traffic classes. The locus of these allocations forms the tradeoff plot. Figure 4.2 shows a hypothetical tradeoff plot for a single VBR call. In order for it to be possible to trade one resource for the other, while keeping the QoS constant, the shape of the tradeoff plot is restricted to curves where increasing the allocation of one resource makes it possible to decrease the amount allocated of the other (as in figure 4.2). Curves of the type $xy = c$ (hyperbola) or $x + y = c$ (straight line) satisfy this requirement.

To obtain the tradeoff plot, the VBR delay and loss rate for a given buffer size and service rate can be calculated using, for example, the analytical model in [4][1]. The minimum service rate that satisfies the QoS can be found by iteration. The tradeoff plot can also be calculated using the effective bandwidth schemes described in [15, 31].

When the size of the queue is small, a large bandwidth allocation—close to the peak rate (point A)—is adequate to satisfy the delay and cell loss requirements. However, as more buffers become available, the bandwidth requirements to satisfy the QoS reduce till, when the buffer size is infinitely large, service at the average rate will still enable the QoS demand to be met. In practice, VBR traffic has a maximum delay/delay jitter bound that will restrict the buffer size to a maximum value, S_{max}. The bandwidth corresponding to this buffer size on the tradeoff plot (point C), is termed as the effective bandwidth, ***Eb***. The static scheme used to calculate the tradeoff plot would use this value as the bandwidth requirement of an incoming VBR call.

---

[1] The model predicts the queue overflow probability alone. However, we have extended it to predict average delay as well ([30]). VBR traffic is assumed to be generated according to an *On/Off* model.

Points on the tradeoff plot correspond to resource allocations that enable the QoS to be satisfied. Points above the plot indicate an allocation of resources that give the VBR call a better QoS than required, while those under the plot indicate that the resources are inadequate to achieve the specified QoS.

The tradeoff plot is actually composed of the intersection of two plots. At high bandwidths and low buffers, the average delay is lower than the QoS specification. As a result, the steep upper portion of the plot depicts the locus of points with bandwidths barely satisfying the required CLR. At large buffer sizes, the cell loss rate of the traffic is better than the required CLR. As a result, the flat portion of the curve is the locus of points with bandwidths just satisfying the delay QoS. The tradeoff plot is obtained by taking the maximum of the two plots at each buffer size.

In what follows, we describe the tradeoff plots generated by two simulation-based cell delay and loss rate models. A neural network model was developed by training an adaptive logic network or ALN [6] on the simulation data (section 2.4). The second model was developed by a regression analysis of the simulation data (section 2.5).

The VBR traffic used to obtain the models was generated according to an On/Off model with exponential on and off times and constant inter-arrival time during a burst. The mean burst length was 353 cells and the average-to-peak ratio was 0.42. The models are computationally inexpensive as compared to the analytical model of [4] and are more accurate—especially at low to medium buffer sizes and high cell loss rate. However, they are not as general as the analytical models—being restricted to *homogeneous* traffic of approximately the same burst length and average-to-peak ratio as in the simulation experiments.

Four QoS levels were used to generate the tradeoff plots in table 4.1[2]. Since the

	Low delay	*High delay*
Low cell loss rate	<0.005 sec, 0.00001>	<0.1 sec, 0.00001>
High cell loss rate	<0.005, 0.001>	<0.1 sec, 0.001>

Table 4.1: QoS levels for tradeoff plots

burst length is 353 cells, the minimum buffer size was set to 400 cells. Each plot consists of three different curves with varying number of multiplexed sources. The bandwidth requirement was normalized w.r.t. the sum of the peak rates of the calls.

The plots shown below for the different QoS levels have some similar properties. For any QoS requirement, it can be seen that the ALN and regression curves are quite similar. For any buffer size and QoS level, the bandwidth requirement decreases as the number of multiplexed calls increases, showing the benefit of statistical multiplexing.

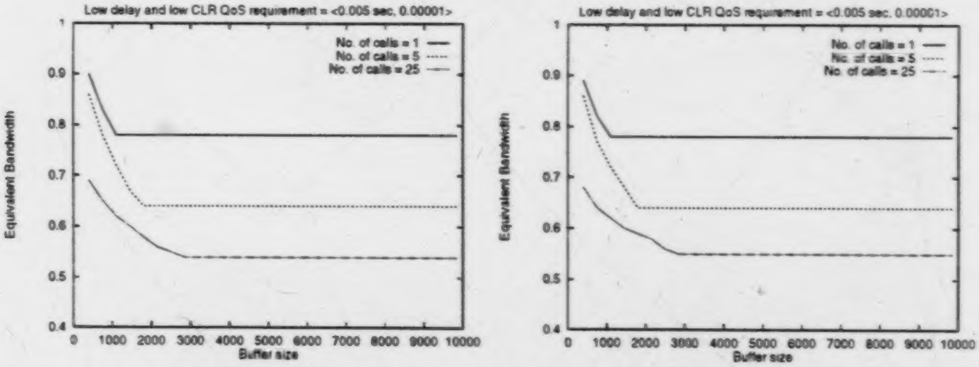


Figure 4.3: Tradeoff plot : regression (left) ALN (right)

Figure 4.3 shows the tradeoff plots for low delay and cell loss. To explain the nature of the plots, we conjecture that the delay is governed primarily by the service rate, while the cell loss is mainly governed by the buffer size. This is in accordance with the previous observation that the portion of the tradeoff plot at high buffer sizes arises from delay requirements, while the steep section at low buffer sizes is a result of the cell loss QoS. When the traffic is generated by a single call, the low delay requirement necessitates a minimum bandwidth of about 0.8 (causing the plot to flatten out at this level). This service rate requires a minimum buffer size of about 1000 cells to meet the low CLR requirement.

---

[2] The maximum delay was not specified since we are only interested in the shape of the tradeoff plots.

However, when the number of calls increases to 25, the minimum bandwidth requirement decreases to 0.57 as a result of statistical multiplexing. This necessitates a larger minimum buffer requirement of about 2800 cells to meet the low CLR, accounting for the rightward shift of the knee of the curve with increasing degree of multiplexing. When we increase the cell loss bound (figure 4.4), while keeping the delay requirement fixed, the minimum buffer requirement decreases below 1000 cells. The minimum bandwidth requirement for a given degree of multiplexing remains the same, as the delay requirement is unchanged. This explains why the knee of the curves in this figure is to the left of the curves in figure 4.3 as well as why the curves flatten out in both figures at the same service rates.

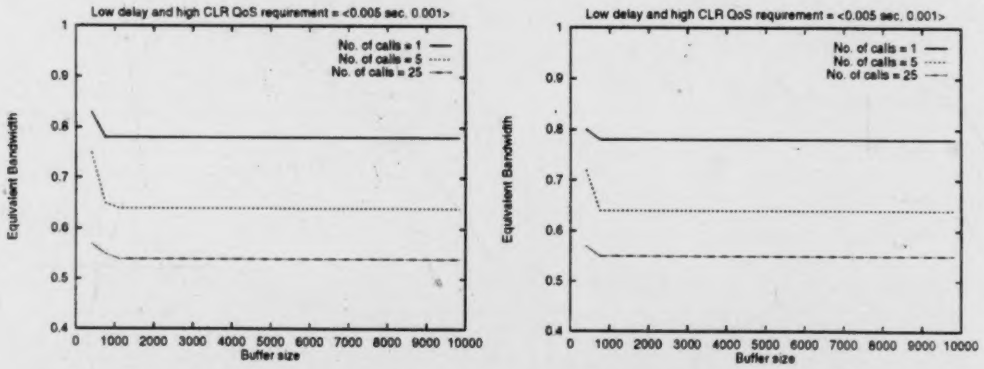


Figure 4.4: Tradeoff plot : regression (left) ALN (right)

Figure 4.5 shows the tradeoff plots for high delay and low cell loss. Relaxing the delay bound allows the minimum service requirement to decrease. Because of this, the minimum buffer requirement moves to the right as compared to figure 4.3 (about 5000 cells). The curves tend to merge at high buffer sizes: when higher delays can be tolerated, the service rate can be lowered to a greater extent, even when few calls are multiplexed. This figure shows that the service rate for all three curves can be lowered to the same minimum level (given by delay requirements)—in spite of the differing degrees of multiplexing.

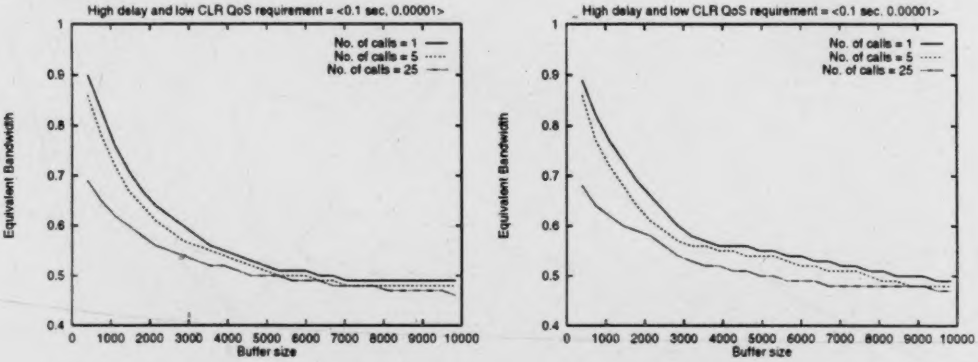


Figure 4.5: Tradeoff plot : regression (left) ALN (right)

Figure 4.6 shows the tradeoff plots for high delay and cell loss. The only effect of raising the cell loss rate bound as compared to the previous plots is to move the knee of the curves to the left, because of lower buffer requirements.

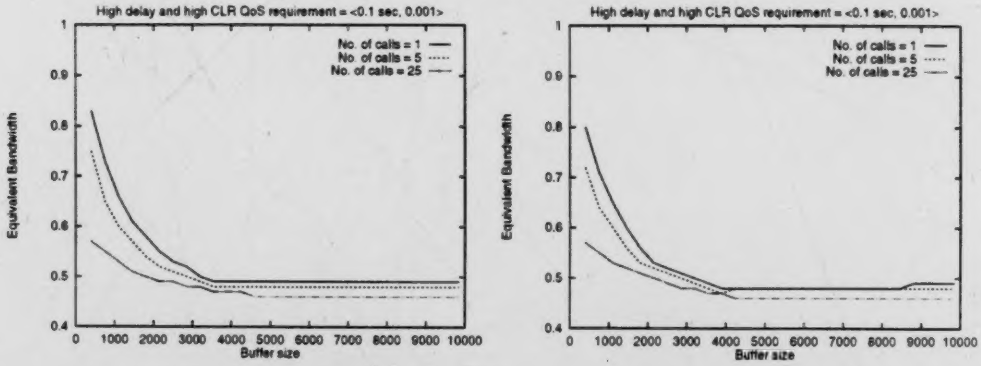


Figure 4.6: Tradeoff plot : regression (left) ALN (right)

The tradeoff plot suggests a strategy for improving the call blocking rate for VBR traffic. Assume that initially there are no VBR or ABR calls in progress. When the first VBR call arrives, a static equivalent bandwidth scheme would allocate the VBR queue

Eb units of bandwidth. Suppose that we then see a flurry of ABR calls, occupying the remaining link bandwidth, $L - Eb$.

One drawback of the static CAC approach that now becomes noticeable is that the bandwidth chosen for the VBR call is based on the maximum buffer size allowable. Since the tradeoff plot curves downward quite sharply, for most maximum delay constraints, the operating point C will be located on the flat portion of the tradeoff plot (figure 4.2)—where the average delay is effectively the only requirement to be met. In this region, the cell loss rate is lower than the CLR requirement, as mentioned earlier.

It would have been better, however, to allocate a larger portion of the bandwidth to the delay-sensitive source so as to place the operating point in the steep portion of the tradeoff plot (point A/B, figure 4.2)—providing lower average cell delay than the requirement—and choosing a buffer size that maintains the cell loss rate at the requisite level.

A second VBR call would have to be refused in these circumstances, because the bandwidth has been mostly utilized by ABR traffic. This situation arises because ABR and VBR traffic are treated with equal priority by the static CAC. As mentioned in section 4.1, we expect the network to profit by giving VBR traffic greater importance at call admission time.

With our proposed scheme, the operating point for the first VBR call is located in the high bandwidth-low buffer region (point A or B, depending on the amount of available bandwidth) with a bandwidth allocation of *Bw* instead of *Eb*. The extra bandwidth $Bw - Eb$ becomes unavailable to new ABR calls. When the second VBR call arrives, the first call can relinquish the additional bandwidth, moving its operating point lower along the tradeoff plot—at worst to point C—to accommodate the new call. Thus, the call blocking rate for VBR traffic is improved by making VBR sources "greedy" in their bandwidth requirements.

It should be pointed out that with a reasonable scheduling mechanism that allocates bandwidth unutilized by VBR traffic to the ABR queue (chapter 3), the extra allocation of bandwidth to the first call does not harm ABR cell traffic—it merely reduces the number of ABR calls. The ABR calls already accepted can send traffic above the minimum allocated rate to absorb the excess bandwidth. Until the second VBR call arrives, the extra bandwidth allocated to the VBR queue is used to improve the average delay of the first VBR call.

4.4 Dynamic CAC

One can expect that with the dropping cost of memory, switch buffers cease to be a precious resource. Moreover, it is far easier to reconfigure a switch with additional memory than it is to install a faster switch along with a link capacity upgrade. Also, sharing a common buffer pool between the ABR and VBR traffic and yet keeping them logically independent may prove too complicated to implement in hardware.

We would like to retain the advantages of dynamically relocating the operating point, without the need to trade cell buffers back and forth between the two queues. If we fix the buffer size of the VBR queue at S_{max}, then allocating peak bandwidth to the VBR call would lower its average delay *as well as* its cell loss rate. This is obviously an unnecessary over-allocation, as VBR traffic is not as sensitive to cell loss as it is to delay and delay jitter.

To resolve this difficulty, one can employ a feedback scheduler (chapter 3) for the VBR queue in order to maintain the VBR QoS at the required level. The service rate of the VBR queue is modulated around a nominal allocation made by CAC in order to reduce the effect of traffic fluctuations and traffic parameter variations on the QoS. The WFQ mechanism guarantees the ABR traffic its share of the bandwidth; as before, any remaining bandwidth can be utilized by the ABR queue.

The usefulness of this scheduling mechanism stems from the fact that irrespective of the service rate allocated by CAC, the cell loss rate is maintained at the requisite level. The service rate allocation only affects the average delay, as shown in

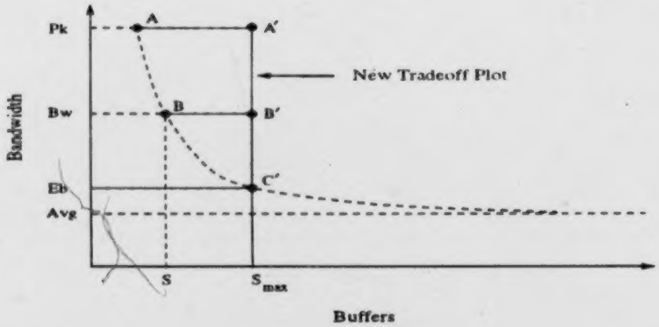


Figure 4.7: Tradeoff plot for a VBR call (Fixed buffer size)

section 3.9.4.

This means that CAC can make any service rate allocation and know that the cell loss rate will be close to the requirement. Allocating close to peak rate will lower the average delay while a service rate close to effective bandwidth will result in an average delay close to the QoS specification. Thus, the combination of D-CAC and feedback control scheduling allows us to lower the call blocking rate and average delay for VBR traffic, without the buffer manipulation entailed by use of the tradeoff plot.

Figure 4.7 shows the new tradeoff plot along which the operating point is varied. The operating points A' and B' lie above the old tradeoff plot. From section 4.3, we recall that this region denotes an over-allocation of resources for a given QoS. However, because of the feedback control scheduler, the cell loss rate is maintained at the required level. Hence, only the other component of the QoS, the average cell delay, is lowered as a result of the choice of operating point.

4.4.1 The S-CAC algorithm

We select a very simple static CAC scheme as the basis for the D-CAC. The effective bandwidth of incoming VBR calls is assumed to be a known value, *Eb*. The effective bandwidth is assumed to be constant, irrespective of the number of VBR calls being multiplexed—an assumption not generally true. However, we only intend to compare the performance of the D-CAC against that of its prerequisite S-CAC. The accuracy of the S-CAC bandwidth allocation does not affect the relative performance.

Therefore, when a VBR call arrives, the CAC checks whether Eb units of bandwidth are available. If yes, the call is accepted; otherwise, it is rejected. When a VBR call departs, the available bandwidth is incremented by Eb units.

Each ABR call has a minimum bandwidth requirement, Mcr. The call admission procedure for ABR traffic is common to both S-CAC and D-CAC. An ABR call is refused if Mcr units of bandwidth are unavailable. If the call is accepted, the source can generate traffic at a rate above Mcr with the understanding that only best-effort delivery is supported for the excess traffic.

4.4.2 The D-CAC algorithm

We use the following notation:

L	The link rate (cells/sec) at the output port, shared between the VBR and ABR queues.
Pcr	The peak rate of a VBR call in cells/sec.
Scr	The average rate of a VBR call in cells/sec.
S_{max}	The maximum size of the VBR queue, obtained by a consideration of the maximum per-node delay and delay jitter bounds.
Eb	The effective bandwidth of a VBR call in cells/sec, required to satisfy the QoS when the queue size is S_{max}.
Mcr	The minimum bandwidth requirement of an ABR call in cells/sec.
Vbr_Bw	The service rate allocated to the VBR queue. For S-CAC, it is given by the product of the number of VBR calls in progress and Eb.
Abr_Bw	The service rate allocated to the ABR queue. It is given by the product of the number of ABR calls in progress and Mcr.
$Avail$	The portion of link bandwidth currently unused by either queue.
N	The number of VBR calls currently admitted to the queue.
α_{VBR}	The call blocking rate for VBR traffic.
α_{ABR}	The call blocking rate for ABR traffic.

Figure 4.8 shows a flowchart of the D-CAC algorithm. When a call is accepted, the service rate allocated to the VBR queue can be normalized w.r.t. the number of calls

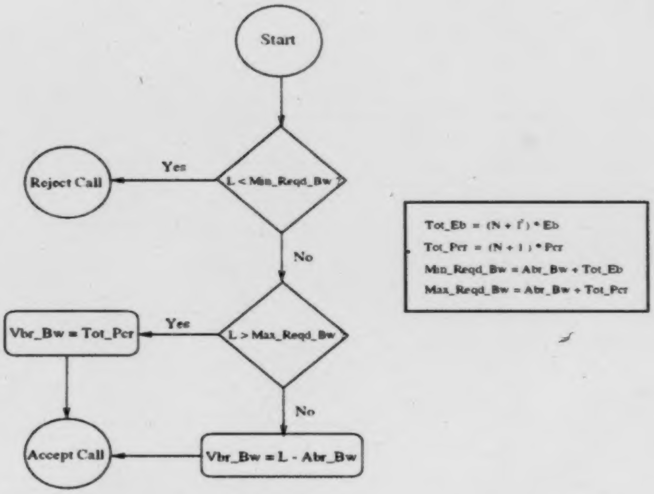


Figure 4.8: D-CAC - incoming VBR call

in progress to obtain the dynamic effective bandwidth. This quantity lies between the static effective bandwidth *Eb* and *Pcr*. In order to compare the static and dynamic CAC, we define the gain G due to D-CAC as

$$G = \frac{Vbr\_Bw}{N * Eb},$$

where Vbr_Bw is the service rate for the VBR queue and N is the number of VBR calls in progress.

As a result of this definition, G lies between 1 and Pcr/Eb. The static CAC has $G = 1$ always, since the bandwidth allocation for a VBR call is always equal to *Eb*. If the average value of G—after a large number of VBR calls have arrived and departed from the queue—is close to Pcr/Eb, it indicates that the D-CAC on average allocated each VBR call a bandwidth close to *Pcr*.

The motivation for introducing G is to allow us to compare the performance of D-CAC and S-CAC, without performing cell-level simulations. As section 4.5.4 shows, when the gain is high, the VBR call blocking rate is much lower than that of S-CAC and, for a work-conserving server, we can conjecture that the higher bandwidth allocation will result in a lower delay.

4.4.2.1 D-CAC on call departure

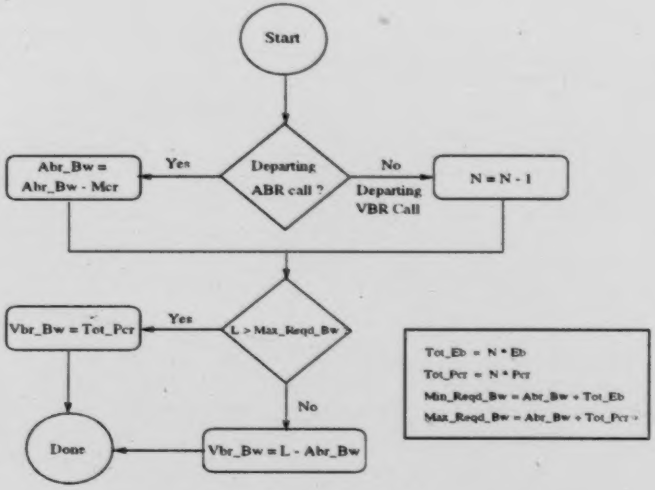


Figure 4.9: D-CAC - departing VBR/ABR call

When a VBR (or ABR) call departs, S-CAC merely decrements the amount of bandwidth allocated to the queue, *Vbr_Bw* (or *Abr_Bw*). However, D-CAC recalculates the amount of bandwidth allocated to the VBR queue at *every* call departure according to the algorithm shown in figure 4.9. Thus, while S-CAC does not allow VBR traffic to benefit by the resources freed by a departing call, D-CAC immediately uses the resources freed up to the advantage of VBR traffic.

4.5 Results

4.5.1 Experiment setup

The performance of the D-CAC was evaluated via call-level simulations. Cell traffic was not generated. Five to ten million calls were simulated in each run of an experiment. To narrow confidence intervals on the collected statistics, each experiment was repeated five times and the statistics were averaged. The following factors were considered in the experiments:

ρ The ratio of the total offered load[3]—due to ABR and VBR—to the link capacity (In the experiments, the link capacity L was fixed at 100 Mbps).

Pcr/L The ratio of the peak cell rate of VBR traffic to the link rate. This determines the maximum number of VBR calls that can be multiplexed using peak rate allocation.

Eb/Pcr The ratio of the static effective bandwidth of a VBR call to its peak cell rate. A small value means that each VBR call will be able to hoard a greater amount of bandwidth, increasing the call blocking rate for ABR traffic and decreasing that for VBR. It also suggests that using D-CAC may result in lower average delays for VBR traffic.

ρ_{abr}/ρ_{vbr} The ratio of the offered ABR load to the VBR load.

The Bellcore document [14] specifies the behavior of test sources used to evaluate the performance of broadband switching systems. VBR test sources are characterized as two-state (On/Off) Markov processes. Three types of VBR test sources conforming to this characterization are shown (table 4.2):

VBR source type	M_a	M_s	P	Scr/Pcr	Pcr/L
VBR I	240	720	6	0.25	0.167
VBR II	500	2500	25	0.16	0.04
VBR III	210	2500	1	0.078	1

Table 4.2: Bellcore VBR test source parameters

The On and Off periods are assumed to be geometrically distributed with mean durations of M_a and M_s cell slots respectively. The cell slot period is the inverse of the capacity (in cells/sec) of the link over which the source conveys traffic to the nearest ATM switch. During an On period, a cell is generated every P cell slots.

---

[3] The offered load due to incoming calls is obtained as the product of the call arrival rate and the bandwidth requirement of each call.

Based on the above, we choose the levels for the factors in our experiments (table 4.3):

Factor	Levels	Default
ρ	0.6, 0.7, 0.8, 0.9	–
Pcr/L	0.04, 0.4	0.04, 0.4
Eb/Pcr	0.2, 0.6	0.2, 0.6
ρ_{abr}/ρ_{vbr}	0.1, 1, 3, 7, 10	1

Table 4.3: Levels of factors used in experiments

From table 4.2, it can be seen that Scr/Pcr of the VBR source types lies in [0.08, 0.25]. Therefore, in choosing the levels for Eb/Pcr, we pick a value close to Scr/Pcr—corresponding to a bandwidth allocation of average rate—and another close to 1—corresponding to near peak rate allocation.

4.5.2 S-CAC : effect of factors on VBR call blocking

In what follows, we show the effect of the above parameters on the call blocking rate for VBR traffic when S-CAC is employed. Figure 4.10 shows the effect of the ratio

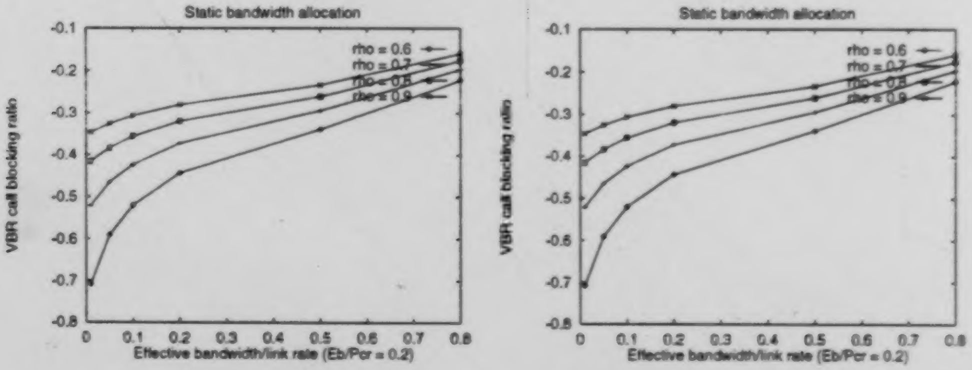


Figure 4.10: S-CAC : effect of Eb/L

$Eb/L = (Pcr/L) * (Eb/Pcr)$, for two values of Eb/Pcr. Pcr itself has no effect on CBR when S-CAC is used. It can be seen that the call blocking rate increases as the effective bandwidth approaches the link rate. This is because, for a given load and call arrival rate, *calls with smaller bandwidth requirement have a higher chance of being accepted*. Also, for calls with small Eb, the load factor ρ has a considerable effect on the call blocking rate. As expected, the Eb/Pcr ratio does not have any effect on the call blocking rate.

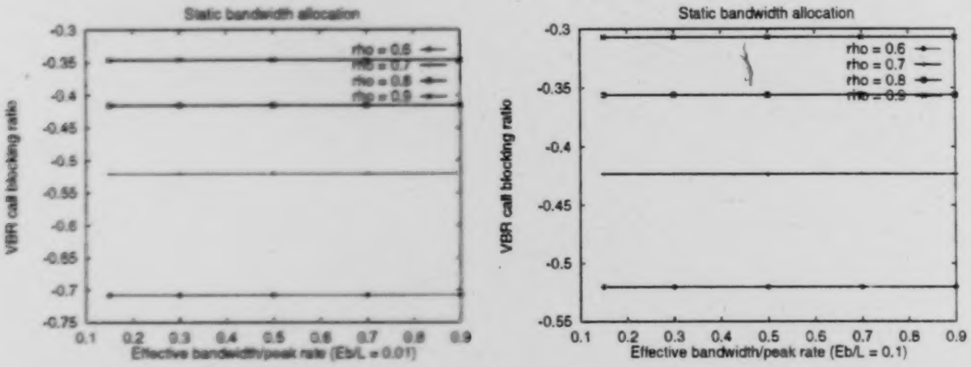


Figure 4.11: S-CAC: effect of Eb/Pcr

Figure 4.11 plots CBR as a function of Eb/Pcr for two values of Pcr/L. As expected, the peak rate has no effect when static CAC is used.

Next, we investigate the effect of ρ_{abr}/ρ_{vbr} on call blocking. In the first set of experiments, Pcr/L was fixed at 0.04 and $Eb/Pcr = 0.2, 0.6$. The experiments were repeated with $Pcr/L = 0.4$. Since Pcr has no effect on the performance of S-CAC, this is equivalent to conducting the experiments with four levels of Eb/L.

Figure 4.12 shows the results for $Pcr/L = 0.04$. When the ratio of load offered by ABR traffic to that offered by VBR traffic is increased—keeping the call arrival ratio fixed—the bandwidth requirement Mcr of an ABR call increases. This increases the call blocking rate for ABR—just as the call blocking rate for VBR increases with Eb in figure 4.10 (left). Therefore, for a given total load, the call blocking rate for VBR decreases as ρ_{abr}/ρ_{vbr} increases.

Figure 4.13 shows the results for $Pcr/L = 0.4$. The conclusions drawn for $Pcr/L = 0.04$ still apply, although the blocking rates are higher due to the higher bandwidth requirement. It can be seen that as Eb approaches the link rate, the call

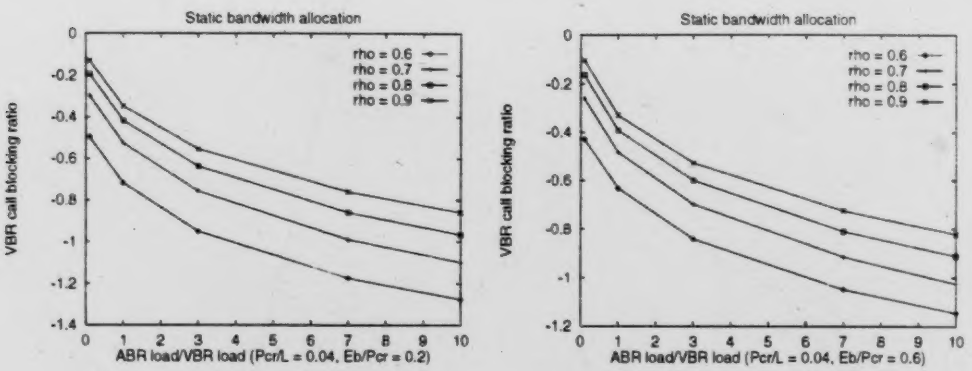


Figure 4.12: S-CAC : effect of ρ_{abr}/ρ_{vbr}

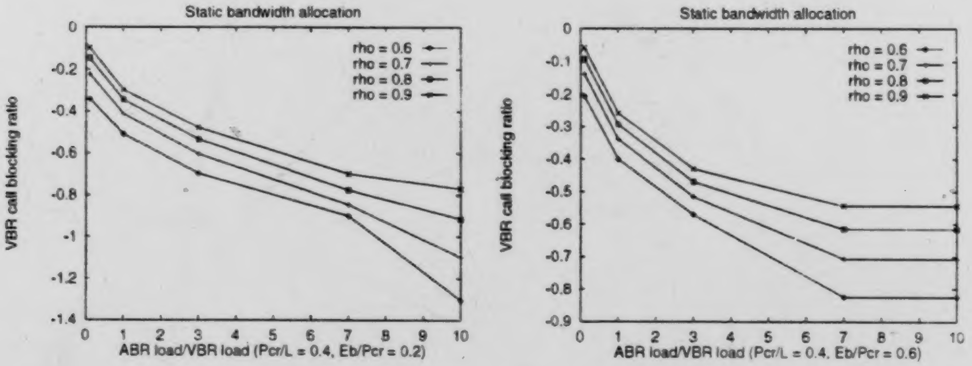


Figure 4.13: S-CAC : effect of ρ_{abr}/ρ_{vbr}

blocking rate flattens off (right).

4.5.3 D-CAC : effect of factors on VBR call blocking

Now, we use D-CAC and compare the effect of the parameters on the call blocking rate for VBR traffic. Figure 4.14 shows the effect of Eb/L. The shape of the plots are the same as for S-CAC, but the D-CAC plots have a sharper knee. This suggests that at high Eb (more than 1/5 of the link rate), there may not be much advantage in using the dynamic scheme.

It is clear by comparison with figure 4.10 that the call blocking rate for low Eb/L is at least an order of magnitude lower with D-CAC. The blocking rate increases with the bandwidth requirement. For calls with Eb close to link rate, the D-CAC call blocking rate approaches that for S-CAC.

Figure 4.15 shows the effect of Eb/Pcr. While the performance of S-CAC was unaffected by the peak rate (for a given effective bandwidth), Eb/Pcr can be seen to have a large impact on the performance of D-CAC, especially at medium loads.

The figure shows that the blocking rate increases with Eb/Pcr. This is due to the following reason: as Eb/Pcr increases, the amount of bandwidth above Eb that can be captured by a VBR call decreases—that is, the degree of greediness of VBR calls is reduced. This results in an increase in the call blocking rate.

Therefore, we can conclude that D-CAC works well when the effective bandwidth requirement is small compared to the link rate and the peak-to-average ratio is high, that is, the traffic is bursty. VBR voice traffic fits these requirements, for example.

Finally, we investigate the effect of ρ_{abr}/ρ_{vbr} on call blocking. As in the case of S-CAC, the first set of experiments are conducted with $Pcr/L = 0.04$ and $Eb/Pcr = 0.2, 0.6$. The experiments are then repeated with $Pcr/L = 0.4$. This is not the same as conducting the experiments at four levels of Eb/L, since the Eb/Pcr impacts the performance of D-CAC (but not S-CAC).

Figures 4.16 and 4.17 show the results for $Pcr/L = 0.04$ and $Pcr/L = 0.4$, respectively. At low Eb/Pcr, the call blocking rate is unaffected by the ratio of ABR load to VBR load (figures 4.16 (left) and 4.17 (left)). At high Eb/Pcr, the call blocking

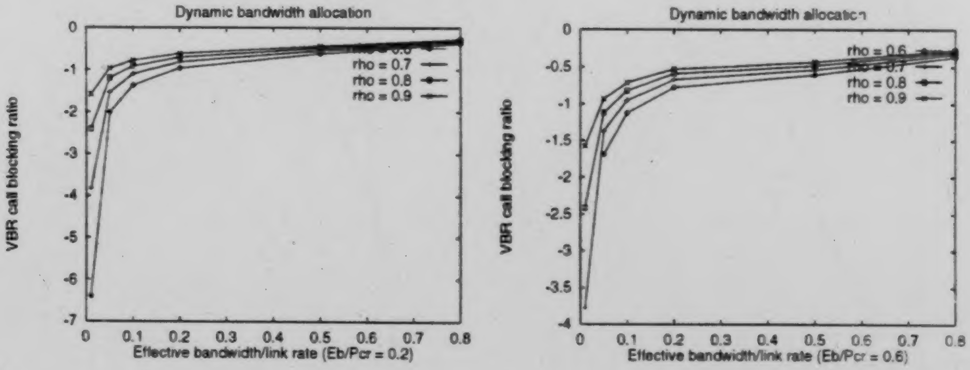


Figure 4.14: D-CAC : effect of Eb/L

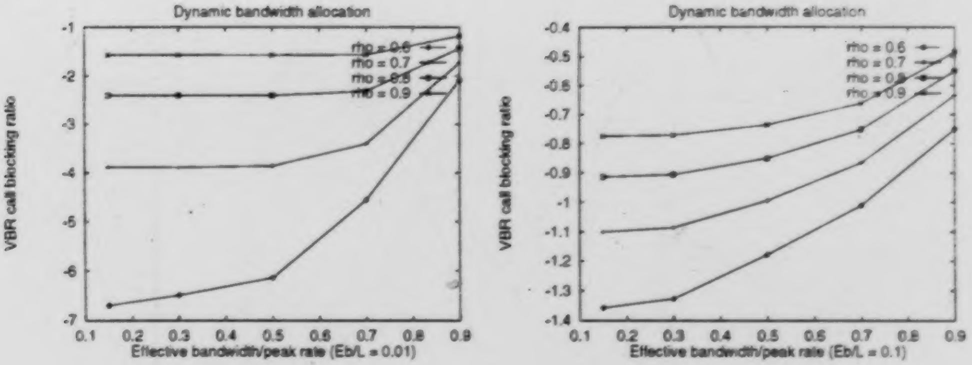


Figure 4.15: D-CAC : effect of *Eb/Pcr*

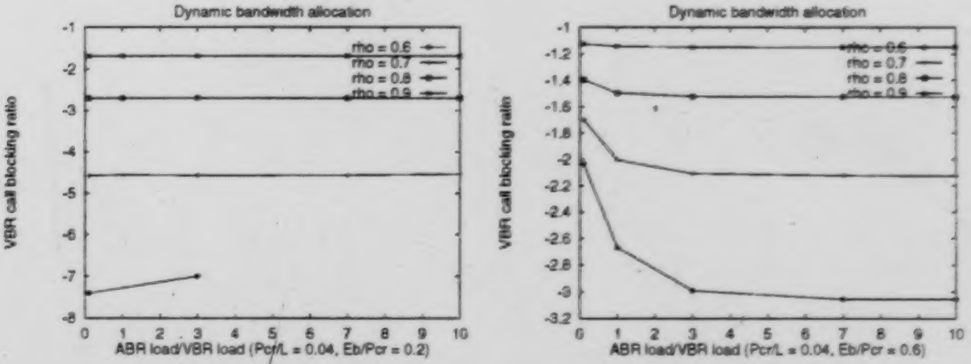


Figure 4.16: D-CAC : effect of ρ_{abr}/ρ_{vbr}

rate increases with ABR load (figures 4.16 (right) and 4.17 (right)) as in the case of S-CAC (the reason for this increase was mentioned in section 4.5.2).

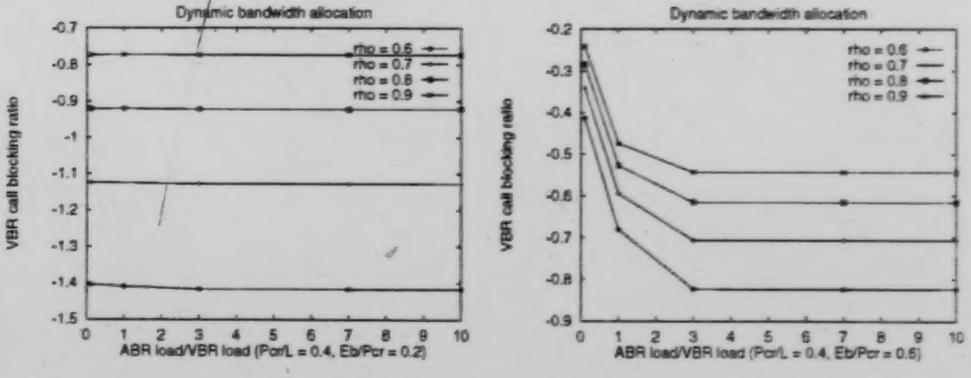


Figure 4.17: D-CAC : effect of ρ_{abr}/ρ_{vbr}

At low Eb/Pcr, bandwidth reservation for a VBR call close to the peak rate results in a large degree of over-allocation (greed). Even at low ρ_{abr}/ρ_{vbr} (or low bandwidth requirement for ABR calls), the high degree of greed causes the ABR calls to be rejected. The only reason in this case for VBR call blocking is because of competition among VBR calls. Therefore, when the ρ_{abr}/ρ_{vbr} ratio increases—resulting in higher bandwidth requirement for ABR calls and correspondingly higher chances of blocking—there is no change in the VBR blocking rate.

At high Eb/Pcr, a reservation close to peak rate does not yield as much benefit—the degree of greediness is small. Hence, when the bandwidth requirement for ABR calls is low, VBR calls are rejected as a result of competition with ABR calls. When the bandwidth requirement for ABR calls increases (due to ABR load increase), the blocking rate for ABR increases leading to an improvement in the blocking rate for VBR.

4.5.4 The modified D-CAC algorithm

Table 4.4 compares the VBR call blocking rate (α_{VBR}) for D-CAC and S-CAC. The calls are assumed to be bursty ($Eb/Pcr = 0.2$) and the degree of statistical multiplexing is assumed to be high ($Pcr/L = 0.01$). The blocking rate for ABR (α_{ABR}) is also shown.

These results indicate that D-CAC is very effective in reducing the call blocking rate for VBR traffic—unfortunately, ABR traffic is very severely impacted. As a remedy, we allow the user to specify the maximum call blocking rate allowable for ABR traffic (α_{ABR}^{max}). D-CAC is expected to improve the call blocking rate for VBR traffic as long

Load ρ	**S-CAC**		**D-CAC**	
	α_{VBR}	α_{ABR}	α_{VBR}	α_{ABR}
0.6	0.2	0.2	3.9997e-07	0.99
0.7	0.3	0.3	1.5e-04	0.99
0.8	0.38	0.38	4e-03	0.99
0.9	0.45	0.45	2.7e-02	0.99

Table 4.4: D-CAC v/s S-CAC

as the call blocking rate for ABR traffic does not exceed the specified value.

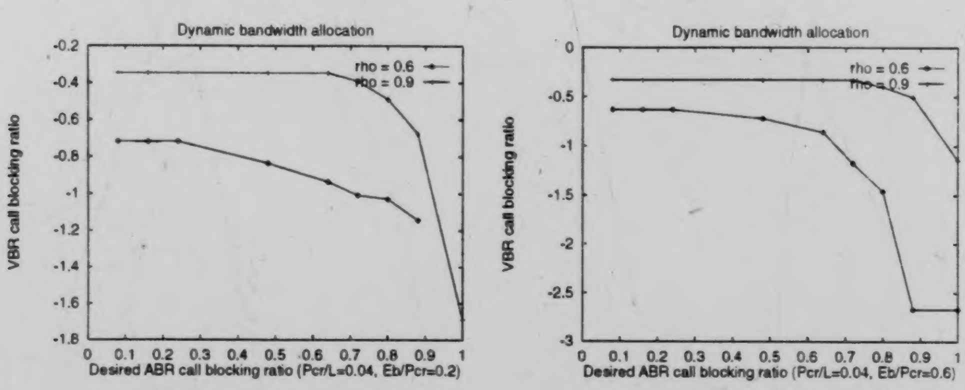


Figure 4.18: MD-CAC : effect of α_{ABR}^{max}

A simple modification to the D-CAC algorithm—the resulting scheme is termed MD-CAC—proved to yield satisfactory results. When the CBR for ABR traffic, α_{ABR}, exceeds $0.8 * \alpha_{ABR}^{max}$ (upper threshold), the maximum dynamic effective bandwidth is set to be equal to *Eb*—equivalently, the dynamic gain $G = 1$—i.e. the queue is allocated a service rate identical to the S-CAC allocation. This conservative allocation allows new ABR calls to be accepted, lowering the call blocking rate for ABR traffic.

When α_{ABR} falls below $0.5 * \alpha_{ABR}^{max}$ (lower threshold), the maximum dynamic effective bandwidth for the VBR queue is set equal to the sum of the peak rates of the VBR calls (dynamic gain $G = Pcr/Eb$)—which means that the MD-CAC is free to allocate any bandwidth between $\sum_{i=1}^{N} *Eb$ and $\sum_{i=1}^{N} *Pcr$ to the VBR queue. This hoarding of bandwidth by the VBR calls may cause new ABR calls to be turned away, raising α_{ABR}.

Thus, the aggressiveness of MD-CAC—in improving the call blocking rate for VBR traffic—is varied in such a way as to ensure that ABR traffic is never affected to an unacceptable extent. We expect the achieved call blocking rate for ABR to lie between 50–80% of the specified maximum. The hysteresis effect caused by the use of two thresholds prevents the operating point on the tradeoff plot from fluctuating rapidly.

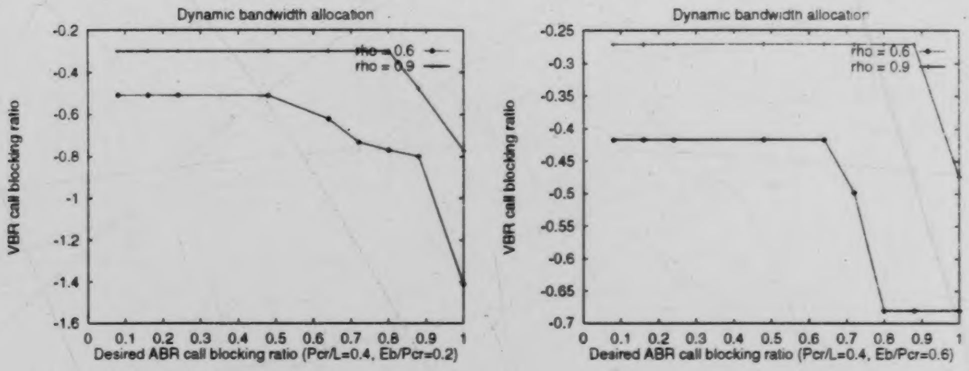


Figure 4.19: MD-CAC : effect of α_{ABR}^{max}

Figures 4.18 and 4.19 show the behavior of MD-CAC as the desired blocking rate for ABR, α_{ABR}^{max}, is varied. The maximum gain is achieved in figure 4.18 (left), showing

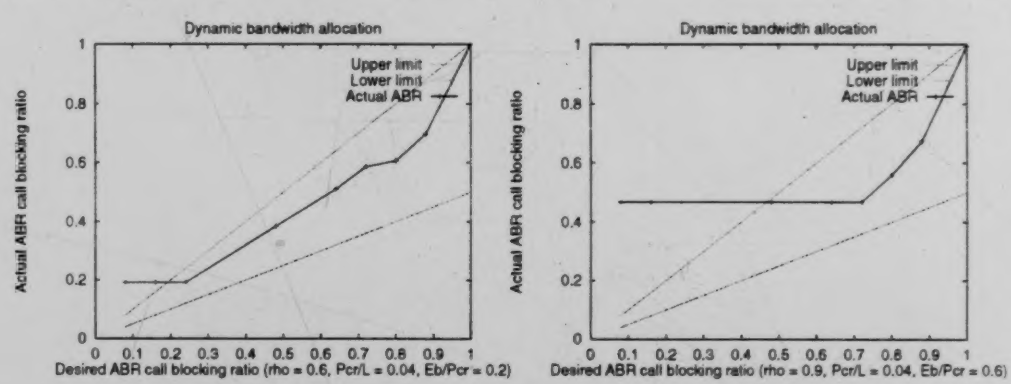


Figure 4.20: MD-CAC : achieved ABR blocking rate

that MD-CAC is most effective for bursty calls with bandwidth requirement much smaller than than link rate. When *Eb* is close to *Pcr*, naturally much benefit cannot be expected from MD-CAC. Figures 4.20 and 4.21 plot the achieved ABR call blocking rate, α_{ABR}, against the desired value, α_{ABR}^{max}. In all cases, the actual blocking rate is seen to be bounded by the upper and lower limits specified.

MD-CAC improves the call blocking rate for VBR at the expense of ABR. Hence, the blocking rate for ABR cannot be lower than that achieved by S-CAC. When the desired blocking rate is set below the S-CAC value, it cannot be achieved. This accounts for the initial flat portion of the curves lying outside the desired limits. Once the desired blocking rate exceeds the minimum value obtained by use of S-CAC, the performance of D-CAC begins to improve, with a corresponding increase in ABR blocking rate.

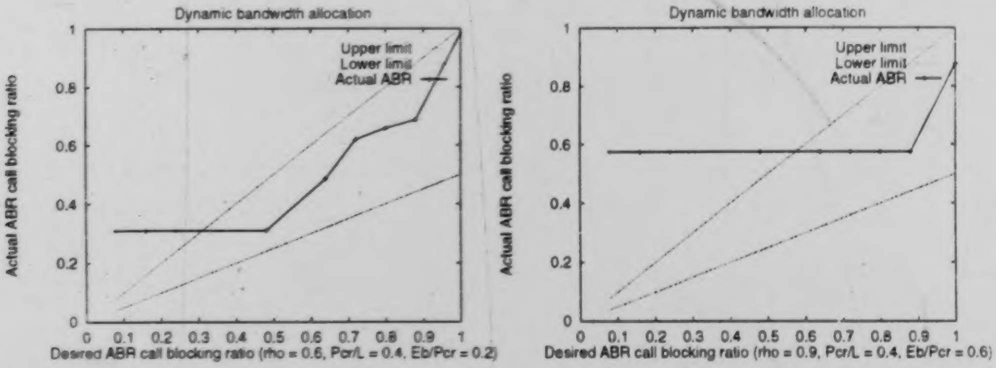


Figure 4.21: MD-CAC : achieved ABR blocking rate

As mentioned earlier, the dynamic gain G allows us to compare the performance of static and dynamic CAC. Figures 4.22 and 4.23 plot G as a function of the desired ABR call blocking rate. Since the dynamic gain normalizes the dynamic effective bandwidth w.r.to Eb, the dynamic gain comparison can only be made for plots with the same Eb. Thus, in figure 4.22, we see that higher gain is achieved for smaller Eb/Pcr (greater burstiness).

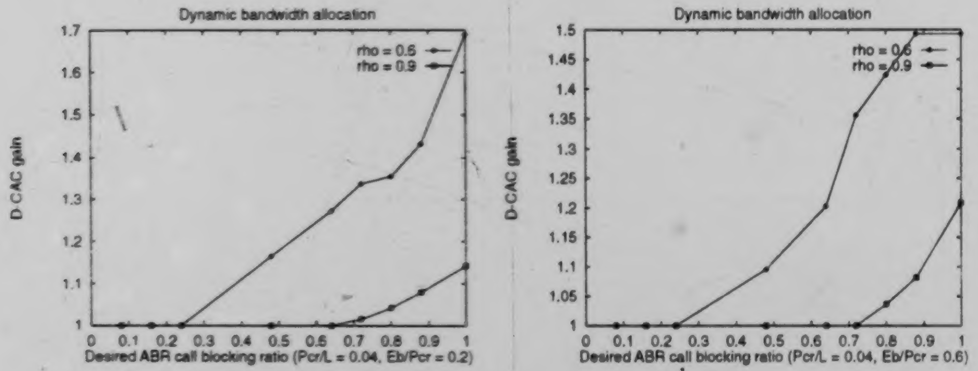


Figure 4.22: MD-CAC : dynamic gain

When $Eb/Pcr = 0.6$ ($Pcr/Eb \approx 1.5$), we see that G reaches the maximum value of 1.5 (for $\rho = 0.6$), indicating that VBR calls are being allocated peak rate on average. In spite of this, the call blocking rate is higher (figure 4.18) than it is for $Eb/Pcr = 0.2$ (In the latter case, the gain is 1.7 against a maximum possible gain of 5 with an average allocation of one-third the peak rate). This shows that it is the gain G, rather than the average allocated bandwidth, that is an index of the improvement in performance as a result of MD-CAC.

From figure 4.22 (left), we see that when $\rho = 0.9$, the maximum gain is 1.1 (at $\alpha^{max}_{ABR} = 1$) which means that the average bandwidth allocated by MD-CAC is $1.1 * Eb$. From the corresponding plot in figure 4.18, we see that the call blocking rate is lower by more than an order of magnitude than in the static case. This shows that, for bursty calls, a small degree of over-allocation in bandwidth can result in a marked improvement in the call blocking rate. Comparing figure 4.22 and figure 4.18, we can see that when the dynamic gain is high, the call blocking rate is correspondingly reduced. A similar conclusion can be drawn from figures 4.23 and 4.19. Thus, the dynamic gain can be used as an indicator of improvement resulting from use of MD-CAC, as noted earlier.

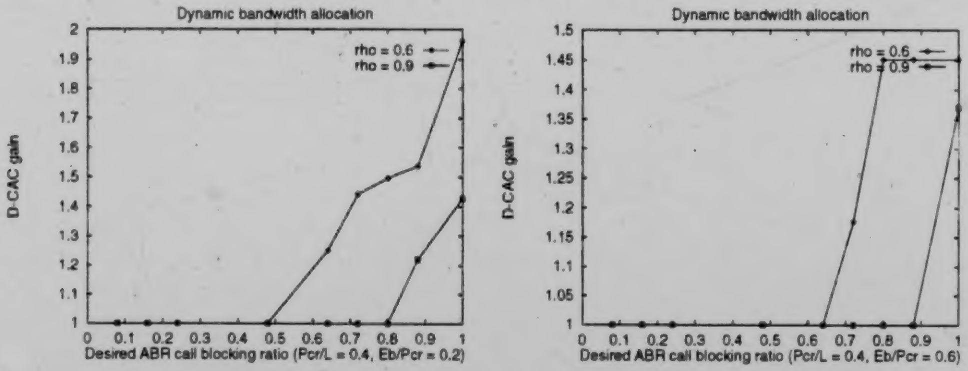


Figure 4.23: MD-CAC : dynamic gain

4.6 Conclusions

We have studied the performance of a dynamic bandwidth allocation scheme for use in CAC for ATM networks. The goal of the dynamic CAC is to improve the call blocking rate for VBR traffic, at the expense of a higher blocking rate for ABR calls (it is expected that VBR traffic will yield greater revenue). The dynamic scheme has the added advantage of improving the average delay for VBR traffic.

We have shown that the dynamic scheme is most beneficial for bursty calls whose individual bandwidth requirements are small compared to the link rate. VBR voice traffic is an important member of this class, even if the link rate is moderate by today's standards. For a high-speed link, this property will hold for all bursty traffic patterns, including video, teleconferencing, etc. The call blocking rate was shown to improve by several orders of magnitude over static allocation in such circumstances.

The impact of the dynamic scheme on ABR blocking can be restricted by specifying the maximum acceptable ABR blocking rate. The actual blocking rate for ABR traffic was shown to lie below this specification in all our experiments.

The dynamic scheme is simple and easy to implement. It can be used in conjunction with any static bandwidth allocation scheme and not just the fixed bandwidth allocation scheme described in this study. Since the dynamic CAC is invoked at call departure as well as call arrival, it allows existing VBR calls to benefit from the resources freed by the departing call—unlike static allocation schemes.

Chapter 5

Conclusions

We have proposed two simple and reasonably accurate schemes for predicting the delay and cell loss when a number of bursty sources are multiplexed at a link with finite buffer space. We have demonstrated the use of the models in accurately predicting the effective bandwidth requirement of each source. We have also shown the need to consider delay bounds—in addition to the cell loss rate requirement—to find the effective bandwidth.

The techniques used to develop the models can be applied to model queuing systems for which analytical solutions are very difficult to develop as a result of complex input traffic, for example, aggregate LAN traffic, MPEG traffic. The models can be developed fairly quickly as compared to analytical models, which is particularly valuable given the diversity of emerging services in B-ISDN networks. They are also computationally inexpensive which makes them suitable for real-time CAC. The difficulty in developing neural network models, in particular, does not increase with the problem dimension. This makes the technique attractive for solving systems with complicated traffic patterns requiring many parameters to characterize.

When the bandwidth prediction by CAC is inaccurate—due to assumptions made during model development or because of incorrect traffic descriptor declaration—the observed QoS for VBR traffic can fluctuate above and below the desired level. The former results in poorer service for ABR, while the latter causes QoS violation for VBR traffic (even though the long-term QoS may be at the required level). Therefore, we have proposed a adaptive feedback control cell scheduler that varies the service rate of the VBR queue to maintain the observed VBR QoS at the specified level, in spite of traffic fluctuations and/or source parameter deviations.

We have demonstrated the ability of the scheduler to control the QoS for three very different aggregate traffic patterns, including self-similar traffic. The performance of the controller was shown to be unaffected by the service rate allocated to the VBR queue by CAC. The initial allocation was shown to impact the average cell delay alone. Therefore, CAC, in conjunction with the feedback control scheduler, can tailor the shape of the buffer distribution to meet source-specific needs.

Finally, in order to increase network earning, we propose a dynamic CAC which lowers the call blocking rate for VBR, assuming that a VBR call is more highly priced than an ABR call (for the same generated traffic) as a result of more stringent QoS requirements. The scheme allocates bandwidth to VBR calls above the requirement estimated by a static CAC such as an effective bandwidth scheme. The extra bandwidth can be traded to accept new VBR calls.

We have shown that over-allocation of bandwidth to the VBR queue does not result in poorer service for ABR, owing to the use of the feedback control scheduler. The average delay, however, can be lower than that with static CAC, depending on the extent of the over-allocation. We also allow the maximum call blocking rate for ABR to be specified, in order to restrict the impact of dynamic CAC on ABR call blocking. We have shown that the actual call blocking rate does not exceed the acceptable value.

Depending on the allowed over-allocation, the dynamic CAC is able to improve the VBR call blocking rate by at least an order of magnitude. The scheme was shown to be beneficial for bursty calls with bandwidth requirement less than one-fifth the link rate, to which category belong voice and video-conferencing traffic.

Chapter 6

Future Work

The VBR traffic that we have considered while developing the regression and neural network models was generated by a number of homogeneous sources. In practice, however, the characteristics of the sources are likely to differ.

It would be interesting to study the degree of heterogeneity allowable before the bandwidth predicted by the assumption of homogeneous sources becomes so inaccurate as to be unusable. Also, some source parameters may have a greater effect on the accuracy of the predicted bandwidth than others. More detailed simulations can be performed to determine ways of aggregating the actual parameter values from multiple sources into a single value that allows the bandwidth to be estimated accurately.

Our studies also assume that the quality of service requirements of the individual sources are the same (homogeneous QoS). When the QoS requirements differ, the most stringent delay and loss rate bounds can be chosen in order to determine the bandwidth required. A study relating the per-source QoS to the overall QoS may have implications for the scheduling policy for cells multiplexed in the same queue and originating from different sources.

The feedback control scheduler varies the service rate for the VBR queue within the CAC specified bounds. It is conceivable that VBR traffic may be multiplexed in more than one queue, for reasons of differing QoS or because of widely differing peak rates. In such a situation, the per-queue feedback controllers must be linked to ensure that the sum of the instantaneous per-queue service rates does not exceed the link rate. A study of the feasibility of controlling all the VBR queues with a single controller can also be made. The performance of the controller with JPEG/MPEG or highly self-similar traffic also needs further study. In addition, the automatic derivation of plant and controller sampling times from the aggregate traffic descriptor needs to be further investigated.

The proposed dynamic CAC can be re-evaluated against more realistic static CACs—especially with schemes whose bandwidth predictions take into account the effect of statistical multiplexing—in order to determine the gain that can be achieved in practice. For example, if the effective bandwidth of a call decreases with increasing number of multiplexed calls, it can be argued than dynamic CAC should perform the same as with sources with small bandwidth requirement—that is, its performance should improve as the number of calls being multiplexed increases. Also, cell-level simulations can be carried out to verify that the average cell delay with dynamic CAC is indeed lower than with static CAC.

Bibliography

[1] R. Addie and M. Zukerman. An approximation for performance evaluation of stationary single server queues. *IEEE Transactions on Communications*, 42(12):3150–3160, 1994.

[2] R. Addie and M. Zukerman. A Gaussian characterization of correlated ATM multiplexed traffic and related queueing studies. *Proceedings of ICC '93*, Geneva, May 1993.

[3] R.G. Addie, M. Zukerman, and T. Neame. Fractal traffic: measurements, moelling and performance evaluation. *IEEE INFOCOM*, 3:977–983, 1995.

[4] D. Anick, D. Mitra, and M. M. Sondhi. Stochastic theory of data–handling system with multiple sources. *The Bell Technical Journal*, 61(8):1871, 1982.

[5] W. W. Armstrong and M. Thomas. Atree 3.0 Educational kit for Windows 3.x, 95, NT. available from http://www.cs.ualberta.ca/~arms.

[6] W. W Armstrong and M. Thomas. *Handbook of Neural Computation – Adaptive Logic Networks(Section C1.8)*. Oxford University Press, 1996.

[7] K.J. Astrom and B. Wittenmark. *Computer Controlled Systems : Theory and Design*. Prentice-Hall Information and System Sciences. Prentice-Hall, 1984.

[8] ATM Forum Technical Committee. Traffic management specification v. 4.0. Technical report, Asynchronous Transfer Mode (ATM) Forum, Available from ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000ps, 1996.

[9] M. Decina and T. Toniatti. On bandwidth allocation to bursty virtual connections in ATM networks. *ICC '90*, 3:844–851, 1990.

[10] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking Research and Experience*, pages 3–26, October 1990.

[11] A. I Elwalid and Mitra D. Effective bandwidth of general markovian traffic sources and admission control of high speed networks. *IEEE/ACM Transactions on Networking*, 1(3):329–343, 1993.

[12] A. Erramilli, O. Narayan, and W. Willinger. Experimental queueing analysis with long-range dependent packet traffic. *IEEE/ACM Transactions on Networking*, 4(2):209–223, April 1996.

[13] P. Gburzynski. *Protocol Design for Local and Metropolitan Area Networks*. Prentice–Hall, New Jersey, 1996.

[14] GR-1110-CORE. Broadband switching system (BSS) generic requirement). Technical Report Issue 1, Bellcore, September 1994.

[15] R. Guerin, H. Ahmadi, and M. Naghsineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE Journal on Selected Areas in Communications*, 9(7):968, 1991.

[16] J. Hyman, A. Lazar, and G. Pacifici. Real–time scheduling with quality of service constraints. *IEEE Journal on Selected Areas in Communications*, 9(7):1052–1063, 1991.

[17] P. Lai. *Transfer Function Modelling : Relationship Between Time-series Variables*. Concepts and techniques in modern geography. 22. Norwich [Eng.] : Geo Abstracts, 1979.

[18] W.E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[19] D. Y. Liu, S. L. Shah, D. G. Fisher, and X. Liu. Multimodel-based minimum bias control of a benchmark paper machine process. *Canadian Journal of Chemical Engineering, to appear*, 1997.

[20] C.L Mallows. Some comments on c_p. *Technometrics*, 15:661–675, 1973.

[21] T. Murase, H. Suzuki, S. Sato, and T. Takeuchi. A call admission control scheme for ATM networks using a simple quality estimate. *IEEE Journal on Selected Areas in Communications*, 9(9):1461, 1991.

[22] Ilkka Norros. On the use of fractional brownian motion in the theory of connectionless networks. *IEEE JSAC '95*, 13(6):953–962, August 1995.

[23] R. Onvural. *Asynchronous Transfer Mode Networks : Performance Issues*. Artech House Inc., MA, 1994.

[24] S. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. PhD thesis, Massachusetts Institute of Technology, February 1992.

[25] V. Paxson. Fast approximation of self-similar network traffic. Technical Report LBL-36750, Lawrence Berkely Laboratory and EECS Division, University of California, Berkely, April 1995. Email : vern@ee.lbl.gov.

[26] V. Paxson and S. Floyd. Wide area traffic : the failure of poisson modelling. *IEEE/ACM Transactions on Networking*, pages 226–244, 1995.

[27] H. G Perros and K. M Elsayed. Call admission control schemes : a review. *IEEE Communications Magazine*, pages 82–91, November 1996.

[28] L. L Peterson and B. S Davie. *Computer Networks : A Systems Approach*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1996.

[29] A Pitsillides and J. Lambert. Adaptive connection admission and flow control : quality of service with high utilisation. *INFOCOM '94*, pages 1083–1091, 1994.

[30] S. Ramaswamy, T. Ono-Tesfaye, W. W. Armstrong, and P. Gburzynski. Equivalent bandwidth characterization for real-time CAC in ATM networks. Submitted to Journal of High Speed Networks, 1997.

[31] K. M. Rege. Equivalent bandwidth and related admission criteria for ATM systems—a performance study. *International Journal of Communication Systems*, 7:181, 1994.

[32] Gunst. R.F and R.L. Mason, editors. *Regression Analysis and its Application—A Data-Oriented Approach*. Marcel Dekker Inc., New York, 1980.

[33] SPSS Inc., Chicago, IL. *SPSS for Windows Advanced Statistics Release 5*, 1992.

[34] E. D. Sykas, K. M. Vlakos, K. P. Tsoukatos, and E. N Protonotarios. Congestion control–effective bandwidth allocation in ATM networks. *IFIP Transactions C [Communication Systems]*, C–14:65, 1993.

[35] F. Vakil. A capacity rule for ATM networks. *GLOBECOM '93*, pages 406–416, 1993.

[36] B. J. Vickers, B. J. Kim, T. Suda, and D. P. Hong. Congestion control and resource management in diverse ATM environments. *IEICE Transactions on Communications*, Nov 1993.

[37] P. E Wellstead and M. B Zarrop. *Self-Tuning Systems : Control and Signal Processing*. John Wiley and Sons, Inc. Sussex, England, 1991.

[38] N. Yin and M. Hluchyj. Simple models for statistically multiplexed data traffic in cell relay networks. *GLOBECOM '93*, 2:824–829, 1993.

[39] L. Zhang. Virtual clock : a new traffic control algorithm for packet switching networks. *Proceedings of ACM SIGCOMM '90*, pages 19–29, September 1990.